МИНИСТЕРСТВО ЭКОНОМИЧЕСКОГО РАЗВИТИЯ И ТОРГОВЛИ РОССИЙСКОЙ ФЕДЕРАЦИИ

«УТВЕРЖДАЮ»	**«УТВЕРЖДАЮ»**
За Министерство экономического развития и торговли Российской федерации	**За Исполнителя**
Заместитель директора Департамента корпоративного управления Минэкономразвития России	**Генеральный директор ООО «Ай-Ферст»**
Ц.В. Церенов	**В.Г. Прижимов**
«____» _____ 2004 г.	**«____» _____ 2004 г.**
«УТВЕРЖДАЮ»	**«УТВЕРЖДАЮ»**
За Исполнителя	**За Исполнителя**
Директор специальных программ и консалтинга ООО «УСП Компьюлинк»	**Генеральный директор ЗАО «РесЭко»**
А.А. Лучин	**Ю.В. Прудкой**
«____» _____ 2004 г.	**«____» _____ 2004 г.**

ОТЧЕТ
О НАУЧНО-ИССЛЕДОВАТЕЛЬСКОЙ РАБОТЕ

ПО ТЕМЕ:

Создание комплекта проектов основных технических нормативных документов по взаимодействию Среды с внешними системами и ИСЭАР

На 410 листах.

Согласовано

_____

_____

_____

«____» _____2004 г.

Согласовано

_____

_____

_____

«____» _____2004 г.

Москва 2004

СПИСОК ИСПОЛНИТЕЛЕЙ

д.ф.-м.н. проф.		В.А. Серебряков
к.ф.-м.н.		А.Н. Бездушный
мл.научн.сотр.		А.М. Меденников
Аспирант		Т.М. Сысоев
Аспирант		А.К. Нестеренко
Сотрудник		А.А. Бездушный
Сотрудник		А.В. Вершинин
к.ф.-м.н.		В.И.Филиппов

РЕФЕРАТ

531 стр., 65 рисунков, 5 таблиц, 12 частей, 29 приложений.

КЛЮЧЕВЫЕ СЛОВА :

Интеграция приложений, сервис-ориентированная архитектура, Web-сервисы, безопасность, модель данных, транзакции, рабочие процессы, глобальная идентификация, метаинформация, адаптеры, среда электронного взаимодействия, электронный административный регламент, автоматизированная система управления взаимодействиями, электронный административный регламент, процедура электронного взаимодействия, профиль стандартов СУВ, эталонная архитектура СУВ, требования к реализации СУВ.

ОБЪЕКТ ИССЛЕДОВАНИЯ ИЛИ РАЗРАБОТКИ

Объектами исследования являются общие возможности внедрения в практику органов исполнительной власти РФ автоматизированных систем управления взаимодействиями (СУВ) - систем интеграции распределенных, гетерогенных АИС и приложений; общие возможности реализации электронных административных регламентов и процедур электронного взаимодействия с помощью АИС «ИСЭАР» и АИС «СЭВ».

ЦЕЛЬ РАБОТЫ

Целью работы является определение основных требований и общих методологических рекомендаций к разработке автоматизированных систем управления взаимодействиями на основе соответствующих технологий и международных стандартов, и как необходимое дополнение - определение основных требований и общих методологических рекомендаций к распределенным, гетерогенным АИС и приложениям, интегрируемым с помощью СУВ, перевод начального набора соответствующих W3C-стандартов.

МЕТОД ИЛИ МЕТОДОЛОГИЯ ПРОВЕДЕНИЯ РАБОТЫ

Анализ существующей ситуации в области систем управления взаимодействиями и интеграции, соответствующих технологий и международных стандартов. Анализ возможностей разрабатываемых СУВ – АИС «ИСЭАР» и АИС «СЭВ», анализ

реализуемых ими электронных административных регламентов (ЭАР) и процедур электронного взаимодействия (ПЭВ).

РЕЗУЛЬТАТЫ РАБОТЫ

Результатами работы является разработка единого подхода к построению автоматизированных систем управления взаимодействиями; общих требований к структуре и содержанию СУВ, общих методологических рекомендаций к разработке СУВ и интегрируемым с их помощью АИС. По указанному направлению выполнены следующие работы:

- Дано описание сложившейся ситуации использования IT-технологий для информационного взаимодействия в WWW сети;
- Рассмотрены общие подходы к проведению интеграции;
- Дано обоснование обеспечения интеграции с применением Web-сервисов (Архитектуры WSA);
- Рассмотрены общие вопросы интеграции, а именно:
    - ИС и сервисы;
    - Транспорт и сообщения;
    - Каноническая модель данных и язык спецификации схем данных;
    - Регистрация сервисов и метаинформации;
    - Безопасность;
    - Транзакции;
    - Рабочие процессы;
    - Глобальная идентификация;
- Изучены общие вопросы интеграции в контексте Web-технологий;
- Даны общие рекомендации по использованию стандартов в рамках СУВ;
- Даны общие рекомендации по использованию стандартов в рамках СЭВ;
- Даны общие рекомендации по использованию стандартов в рамках ИС ЭАР;
- Рассмотрены общие вопросы Интеграция СЭВ с внешними ИС;
- Рассмотрены общие вопросы интеграции ИС ЭАР с внешними ИС;
- Рассмотрены общие вопросы интеграция СЭВ с ИС ЭАР.

Осуществлены переводы следующих стандартов W3C консорциума:

- SOAP 1.1,
- XML_Схемы_Типы_Данных,
- XML_Схемы_Структуры,

- WSDL 1.1,
- BPEL4WS_1.1,
- UDDI_2.03_Data_Structure,
- UDDI _2.04_API_Specification.

Приведены схемы типовых данных:

- XML-схема оргсправочника НСИ ИС ЭАР,
- Документация по XML-схеме оргсправочника НСИ ИС ЭАР,
- Пример XML-файла с данными по к XML-схеме оргсправочника НСИ ИС ЭАР,
- XML-схема для представления контролируемых словарей в НСИ ИС ЭАР,
- Документация по XML-схеме для представления контролируемых словарей в НСИ ИС ЭАР,
- Пример XML-представления классификатора в НСИ ИС ЭАР.

ОБЛАСТЬ ПРИМЕНЕНИЯ

Федеральные и региональные органы исполнительной власти.

ЭКОНОМИЧЕСКАЯ ЭФФЕКТИВНОСТЬ ИЛИ ЗНАЧИМОСТЬ РАБОТЫ

Определение основных требований и методологических рекомендаций к разработке автоматизированных систем управления взаимодействиями базируется на следование набору международных стандартов. Методологические рекомендации, соответствующие международным стандартам, образуют технологическую базу для разработки АИС «ИСЭАР» и АИС «СЭВ», интеграции распределенных, гетерогенных АИС и приложений, реализации электронных административных регламентов и процедур электронного взаимодействия, которые обеспечивают технологический базис преобразований в рамках административной реформы и реформы государственной службы.

СОДЕРЖАНИЕ

ТАБЛИЦЫ

РИСУНКИ

НОРМАТИВНЫЕ ССЫЛКИ

В настоящем отчете о НИР использованы ссылки на следующие государственные стандарты:

1. ГОСТ Р 1.0-92. Государственная система стандартизации Российской Федерации. Основные положения. Введен 1993-01-01
2. ГОСТ Р 1.2-92. Государственная система стандартизации Российской Федерации. Порядок разработки государственных стандартов. Введен 1993-01-01
3. ГОСТ Р 1.5-92. Общие требования к построению, изложению, оформлению и содержанию стандартов. Введен 1 июля 1992 г.

ОПРЕДЕЛЕНИЯ

Используемые в настоящем документе термины и основные понятия области Интернет понимаются в соответствии с действующими рекомендациями (RFC – Request for Comments – рабочие предложения) международных органов, ответственных за вопросы стандартизации в Интернет.

Термины, используемые при описании ролей пользователей и процессов функционирования подсистем, определены в спецификации UML (Unified Modeling Language – универсальный язык моделирования).

Используемые в Документе термины и основные понятия области автоматизированных систем определены в ГОСТ 34.003-90.

Принятые сокращения :

АИС – автоматизированная информационная система

АРМ – автоматизированное рабочее место

АРП – автоматизированный рабочий процесс

АС – автоматизированная система

БД – база данных

ГИР - государственных информационных ресурсов

ИА – интерфейс администратора

ИКТ - информационно-коммуникационные технологии

ИС – информационная система

ИСЭАР– исполняющая система ЭАР

ИТЦ – информационно–телекоммуникационный центр

ИП – интерфейс пользователя

НСД – несанкционированный доступ

НСИ – нормативно-справочная информация

ОГИР – объединение государственных информационных ресурсов с целью их совместного использования.

ОС – операционная система

ПО – программное обеспечение

ПТК – программно-технический комплекс

ПЭВ – процедура электронного взаимодействия

РД – рабочая документация

РДИ – регистр деловой информации региона

СВТ – средства вычислительной техники

СКЗИ – специализированный комплекс защиты информации

Среда (СЭВ) - Среда электронного взаимодействия федеральных органов исполнительной власти и хозяйствующих субъектов

ССК – справочники, словари и классификаторы

СУБД – система управления базами данных

СУВ – система управления взаимодействием

СЭДО – система электронного документооборота

ТЗ – техническое задание

ТП – технический проект

УЦ – удостоверяющий центр

УЭПИ – управляющий элемент пользовательского интерфейса (экранная кнопка, гиперссылка, пункт меню и т.п.).

ФОГВ – федеральный орган государственной власти

ФЦП – Федеральная целевая программа

ХС – хозяйствующий субъект

ЧТЗ – частное техническое задание

ЭАР – электронный административный регламент

ЭД – эксплуатационная документация

ЭЦП – электронная цифровая подпись

API – Application Programming Interface – интерфейс программирования приложений

BPEL4WS – Business Process Execution Language for Web Services – язык исполнения бизнес-процессов

DTD – Document Type Definition – Определение Типа Документа, регламентированное XML 1.0

EDI – Electronic Data Interchange – электронный обмен данными

ER – Entity-Relationship model – модель сущность-отношение

G2B – модель взаимодействия органов власти с хозяйствующим субъектом

HTML – Hypertext Markup Language – гипертекстовый язык разметки

HTTP – Hypertext Transfer Protocol – протокол передачи гипертекста

ODMG – Object Database Management Group – группа управления объектными базами данных

OWL – Web Ontology Language – язык Web-онтологий

RDF – Resource Description Framework – каркас Описания Ресурсов

RDFS – Resource Description Framework Schema – схема описания структуры ресурсов

RDF/XML – XML-синтаксис для RDF

RFC – Request for Comments – рабочие предложения

SOAP – Simple Object Access Protocol – простой протокол взаимодействия с объектами

SQL99 – Structured Query Language 99 - язык структурированных запросов версии 99

UDDI – Universal Description, Discovery, and Integration – глобальное описание, поиск и взаимодействие

UML – Unified Modeling Language – универсальный язык моделирования

URI – Uniform Resource Identifier - универсальный идентификатор ресурса

URL – Uniform Resource Locator – универсальный указатель расположения ресурса

W3C – World Wide Web Consortium – WWW-консорциум

WSDL – Web Service Definition Language – язык описания web-сервисов

XHTML – Extensible Hypertext Markup Language – расширяемый гипертекстовый язык разметки

XML – eXtensible Markup Language – расширяемый язык разметки

XLANG/s – язык описания деловых процессов в Microsoft BizTalk Server

XMI – XML Metadata Interchange – формат XML-обмена метаданными

XSD – документ XML Schema (XML-схема)

1 Введение

1.1 Основание для разработки

Разработка ведется на основании следующих документов:

1. Государственный контракт № ЭР.03.16 от 21.07.04 «Развитие процедур электронного взаимодействия органов государственной власти и хозяйствующих субъектов», заключенный между Министерством экономического развития и торговли Российской Федерации и ООО «Ай-Ферст»
2. Государственный контракт № ЭР.03.17 от 09.06.04 «Разработка и внедрение электронных административных регламентов на примере структурных подразделений Минэкономразвития России», заключенный между Министерством экономического развития и торговли Российской Федерации и ООО «УСП Компьюлинк»

По Государственному Контракту N ЭР.03.16 от 21.07.04. разработаны следующие разделы документа:

- Разделы 1, 2, 3, 4, 5, 6, 7
- Раздел 10 -в части описывающей СЭВ (подразделы 10.3, 10.4)
- Приложения 13.1 "Переводы стандартов"
- Приложения 13.4 "Спецификация административных интерфейсов СЭВ"

По Государственному Контракту N ЭР ЭР.03.17 от 09.06.04. разработаны следующие разделы документа:

- Разделы 8, 9
- Раздел 10 -в части описывающей ИС ЭАР (подразделы 10.1, 10.2)
- Приложения 13.2 "Схемы типовых данных"
- Приложения 13.3 "Спецификация административных интерфейсов ИСЭАР"

1.2 Цели разработки и назначение технических нормативных документов по интеграции.

Целями разработки технической нормативной документации являлись:

1. Определение состава основных технических нормативных документов по интеграции СЭВ и сторонних ИС, ИС ЭАР и сторонних ИС, СЭВ и ИС ЭАР для обеспечения наиболее эффективного взаимодействия.
2. Разработка общих рекомендаций по применению ТНД при интеграции СЭВ со сторонними ИС, ИС ЭАР со сторонними ИС, СЭВ и ИС ЭАР.

1.3 Целевая аудитория

Настоящий документ ориентирован на руководителей и технических специалистов, информационных и технологических служб Федеральных и Региональных органов власти, органов власти субъектов Федерации и органов местного самоуправления.

1.4 Структура документа

Разработанный документ предназначен для использования в качестве руководства по интеграции ИС с СЭВ и с ИС ЭАР, а так же в качестве справочного руководства по общим вопросам проведения интеграции ИС в сети Интернет, в частности, подробно рассмотрены вопросы на базе WSA. Документ имеет следующую структуру:

В разделе 1 настоящего документа раскрыты цели разработки документа, целевая аудитория документа, дано краткое описание IT технологий для информационного взаимодействия в WWW сети. Показаны общие подходы к проведению интеграции между ИС и выделена роль СУВ как базового подхода к интеграции.

В разделе 2 настоящего документа рассмотрены общие подходы к проведению интеграции между ИС на базе СУВ и обоснованы группы механизмов обеспечивающих решение задачи интеграции:

- ИС и сервисы;
- Транспорт и сообщения;
- Каноническая модель данных и язык спецификации схем данных;
- Регистрация сервисов и метаинформации;
- Безопасность;
- Транзакции;
- Рабочие процессы;
- Глобальная идентификация.

Для каждой группы приведены критерии и определены требования.

В разделе 3 настоящего документа рассмотрены варианты построения решений по интеграции на базе архитектур WSA, CORBA, DCOM. Произведен сравнительный анализ решений по основным группам механизмов выделенных в разделе 2 и выбрана оптимальная архитектура WSA.

В разделе 4 настоящего документа в разрезе выбранной архитектуры WSA приведено описание поддерживаемых стандартов в разрезе групп определенных в разделе 2. Произведен выбор оптимального стандарта для каждой группы механизмов – «базисного» стандарта в рамках СУВ построенной на базе WSA.

В разделе 5 настоящего документа подробно рассмотрены выбранные в разделе 4 стандарты. Выявлены узкие места выбранных стандартов и приведены рекомендации по их устранению для каждой из групп, в рамах СУВ. Приведены общие рекомендации по применению выбранных стандартов для интеграции в рамках СУВ.

В разделе 6 настоящего документа приведены рекомендации и особенности применения выбранных стандартов для использования в рамках ИС СЭВ. Рассмотрены вопросы доработки существующих стандартов ИС СЭВ и рекомендации по использованию стандартов СУВ, которые в ИС СЭВ не реализованы. Представлен перечень неиспользуемых в ИС СЭВ «базисных» стандартов и приведены решения по целесообразности доработки ИС СЭВ до возможности работы с выбранным «базисным» стандартом для каждой из групп.

Раздел 7 настоящего документа является методикой проведения интеграции ИС СЭВ с внешними ИС на основе выбранных ранее «базисных» стандартов в рамках WSA архитектуры. Методика построена как пошаговое описание действий, направленных на проведение работ по интеграции. Каждый шаг содержит: четкое описание работ, с привязкой к требованиям, требования к квалификации персонала задействованного в работах, примерной оценки трудозатрат и времени, необходимых для выполнения работ.

В разделе 8 настоящего документа приведены рекомендации и особенности применения выбранных стандартов для использования в рамках ИС ЭАР. Рассмотрены вопросы доработки существующих стандартов ИС ЭАР и рекомендации по использованию стандартов СУВ, которые в ИС ЭАР не реализованы. Представлен перечень неиспользуемых в ИС ЭАР «базисных» стандартов и приведены решения по целесообразности доработки ИС СЭВ до возможности работы с выбранным «базисным» стандартом для каждой из групп.

Раздел 9 настоящего документа является методикой проведения интеграции ИС ЭАР с внешними ИС на основе выбранных ранее «базисных» стандартов в рамках WSA

архитектуры. Методика построена как пошаговое описание действий, направленных на проведение работ по интеграции. Каждый шаг содержит: четкое описание работ, с привязкой к требованиям, требования к квалификации персонала задействованного в работах, примерной оценки трудозатрат и времени, необходимых для выполнения работ.

Раздел 10 настоящего документа является методикой проведения интеграции ИС ЭАР и ИС СЭВ на основе выбранных ранее «базисных» стандартов в рамках WSA архитектуры. Методика построена как пошаговое описание действий, направленных на проведение работ по интеграции. Каждый шаг содержит: четкое описание работ, с привязкой к требованиям, требования к квалификации персонала задействованного в работах, примерной оценки трудозатрат и времени, необходимых для выполнения работ.

1.5 Описание сложившейся ситуации использования IT-технологий для информационного взаимодействия в WWW сети.

Последние несколько лет характеризуются существенными изменениями в деятельности органов власти, связанными с внедрением ИКТ и направленными, в частности, на расширение перечня услуг, основанных на электронном взаимодействии. Началась и успешно продолжается реализация таких программ как ФЦП «Электронная Россия» и ГЦП «Электронная Москва». В связи с увеличением количества услуг, использующих электронное взаимодействие с применением WWW-сети, возникла проблема обеспечения эффективного взаимодействия межу ИС, участвующими в процессах оказания таких услуг.

На сегодняшний день уровень используемых технологий информационного взаимодействия в России не соответствует уровню технологий, используемому во многих развитых странах с более эффективным использованием возможностей WWW-сети.

Во многих случаях информационное взаимодействие между организациями (а также, внутри организаций), между гражданами и организациями реализуется посредством передачи информации на бумажном носителе или посредством телефонных сообщений. В других случаях, даже при применении сети Интернет, в качестве инструмента взаимодействия, технология взаимодействия чаще всего не соответствует современным стандартам.

Взаимодействие в сети Интернет на текущий момент используется на уровне передачи текстовой и графической информации. Сегодня, граждане и сотрудники организаций ежедневно используют сеть для пересылки сообщений, просмотра биржевых котировок, оформления заказов, просмотра сообщений и документов.

Это разнообразие инициатив в области электронизации государства можно распределить по нескольким категориям – от более простых форм взаимодействия к более сложным:

- **Публикация (распространение) информации.** Реализация приложений, которые делают доступной через Интернет информацию в электронной форме. Это пассивное взаимодействие, при котором пользовать не имеет электронного доступа к правительственному ведомству, и, соответственно, ведомство не имеет контакта с пользователем.
- **Электронные формы (и интерактивное взаимодействие).** Реализация приложений, которые предоставляют возможность электронного доступа к правительственному ведомству, для осуществления доступа к информационным услугам. Это активное взаимодействие, когда ведомство имеет возможность электронным способом ответить пользователю.
- **Транзакции.** Реализация приложений, которые предоставляют возможность обслуживания запросов пользователей, инициирующих неделимую последовательность действий разных АИС и сотрудников ведомства. Например, подача заявок в электронной форме на получение лицензий на ведение профессиональной деятельности, подача налоговых деклараций и т.п.
- **Интегрирование.** Это приложения, которые предоставляют новые типы сервиса, ранее невозможные без использования ИТ, обеспечивающие выполнение требуемых неделимых действий вне рамок одной системы или одного ведомства, интеграцию их данных с целью решения комплексной задачи для гражданина. В качестве примера можно представить сервис, который по инициативе гражданина обеспечивает автоматический сбор финансовой информации о гражданине, путем обращений к ИС разных ведомств, возможно, банков, страховых компаний и т.п., на основе полученной информации заполняется специальная форма о финансовых операциях, выплате налогов. В результате гражданину для ознакомления предоставляется готовая декларация о доходах и расходах.

Первыми шагами федеральных, региональных и местных органов власти в предоставлении услуг гражданам являются «средства публикации», реализуемые как

соответствующие сайты или порталы, на которых увеличивается количество «средств интерактивного взаимодействия».

Более реальную дополнительную пользу гражданам в виде значительной экономии времени и средств на взаимодействие с государством могут принести только транзакционные и интеграционные сервисы. Польза для государства будет состоять в повышении качества обслуживания граждан, в экономии средств, которые тратятся на такое обслуживание. В этом случае от такого взаимодействия выигрывают обе стороны. Пока Интернет слабо поддерживает программно-ориентированное взаимодействие систем между собой, систем с гражданами. Гражданам, ведомствам и компаниям, занимающимся как коммерческой, так и иной деятельностью, необходимы средства публикации информации о своих приложениях, услугах и источниках данных, обеспечения их взаимосвязи при решении комплексных задач своих потребителей. Для этого приложения должны иметь возможность средствами Интернет-сети находить другие приложения, обращаться к ним и автоматически взаимодействовать с ними, обеспечивая требуемый уровень конфиденциальности, защиты используемых ресурсов.

Ясно, что транзакционные и интеграционные формы взаимодействия потенциально несут большую ценность, и, конечно, большую технологическую сложность. В публикациях [46] это иллюстрируют следующим образом:

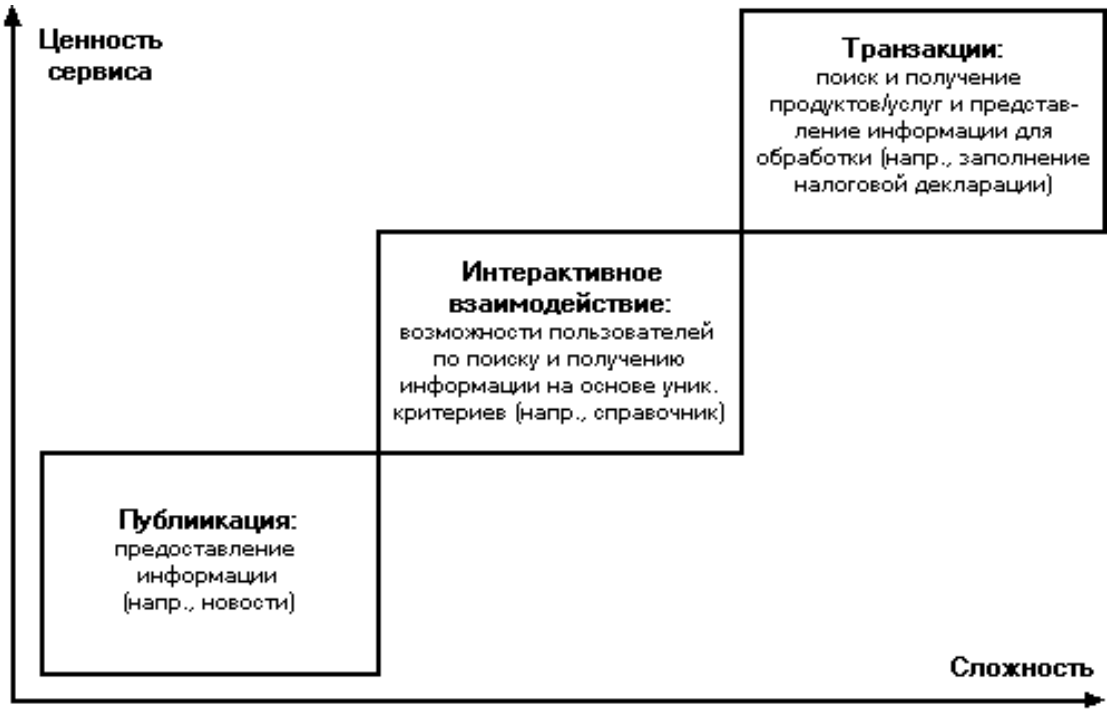


Рисунок 1. Схема взаимосвязи ценности и сложности форм взаимодействия.

При этом отмечается, что новое качество предоставления услуг достигается тогда, когда организации используют новые технологии для реорганизации всего процесса

предоставления услуг, с чем невозможно не согласиться. Указывается, что в отношении государственных структур потенциальный путь (2) может проходить от публикации информации через организацию интерактивного взаимодействия к электронным транзакциям, затем к интеграции государственных услуг одного или разных ведомств, а затем к трансформированию работы правительства и ведомств, к реализации административной реформы.

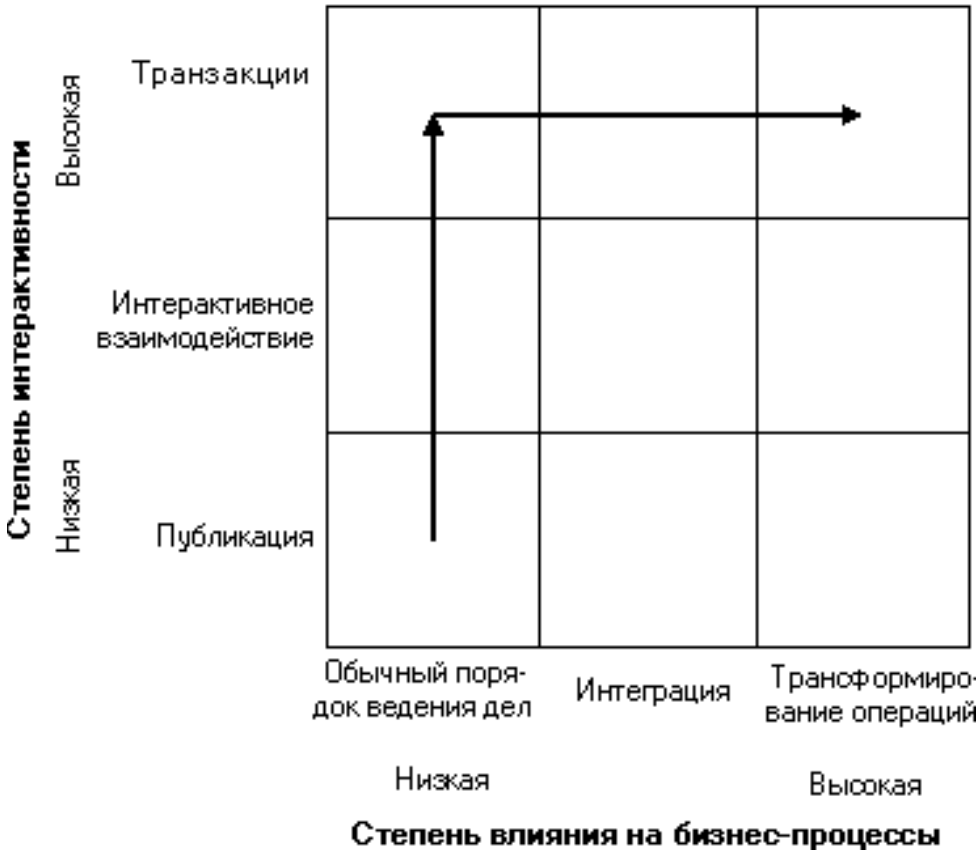


Рисунок 2. Потенциальный путь развития средства предоставления услуг.

Подход к развитию средства предоставления услуг, к построению информационного взаимодействия, которое будет обеспечивать, в первую очередь, взаимодействие между ИС, должен базироваться на разработке единого набора стандартов, оптимального для обеспечения такого информационного взаимодействия. В противном случае, если единый набор стандартов не будет зафиксирован, то возникнут проблемы по интеграции ИС друг с другом (это потребует дополнительных средств), а в ряде случаев, когда произвести интеграцию будет невозможно, то задача будет тупиковой.

1.6 Общие подходы к проведению интеграции

1.6.1 Варианты построения информационного взаимодействия

Интеграция Приложений (ИП) заключается в том, чтобы совместно использовать данные и процессы без необходимости серьезных изменений в приложениях или структурах данных. До последнего времени в основном строились программные системы, предназначенные для одной цели для одного множества пользователей без достаточного продумывания интеграции этих систем в большие системы и многие приложения. Эти системы обычно специально разрабатывались с учетом специфических нужд и использованием текущей технологии. Во многих случаях используются нестандартные хранилища данных и технологии разработки приложений. К сожалению многие из этих систем трудно адаптируются для коммуникаций и использования общей информации с другими, более развитыми системами. В результате возникает проблема интеграции систем и приложений. На первых этапах применения компьютеров информация обрабатывалась на централизованных платформах. В результате процесс и данные существовали в однородной среде. Интеграция приложений (как процессов, так и данных) внутри одной машины редко порождала проблему, не сводящуюся к дополнительному кодированию.

Часто используются одновременно много различных технологий. Интеграция этих технологий - почти всегда трудная задача. Традиционные технологии, ориентированные на передачу сообщений, связывают приложения друг с другом, но эти решения «точка-точка» порождают прямые связи между многими приложениями. В результате поддержка самого решения по интеграции может стать более дорогим, чем поддержка связываемых приложений. При использовании подхода точка-точка интегрируемые приложения должны быть изменены так, чтобы каждое приложение было способно посылать и принимать сообщения. Хотя это легко сделать в случае двух приложений, интеграция дополнительных приложений требует дополнительных каналов. Если приложение A успешно интегрировано с приложением B, и надо включить приложения C и D, нужно создать канал между каждым приложением. Незаметно процесс становится столь сложным, что его реализация становится почти неуправляемой.

Окончательная цель ИП – общая виртуальная система. Это обеспечивает немедленную доступность любой информации, требуемой для всех запросов вне зависимости от того, где информация расположена. Любое приложение, база данных, таблица транзакций, метод и другие элементы вычислений доступны в любое время и

везде. В результате разнообразные системы связываются таким образом, что они выглядят как монолитное и унифицированное приложение.

1.6.1.1Типы ИП

ИП уровня интерфейса к приложению – под этим имеется в виду использование интерфейсов, предназначенных для доступа к приложениям. Используя эти интерфейсы, можно связывать вместе многие приложения, давая им возможность совместно использовать бизнес-логику и информацию. Для интеграции этих систем с другими нужно обеспечить доступ как к процессам, так и к данным, извлечь информацию, преобразовать ее в формат, понятный целевому приложению, и передать информацию. Хотя для этого могут использоваться различные технологии, предпочтительным представляется использование брокеров сообщений.

ИП уровня методов означает совместное использование бизнес-логики. Метод может быть доступен из любого числа приложений, и приложения могут получать доступ к методам друг друга без необходимости переписывания каждого метода внутри соответствующего приложения. Имеется множество механизмов совместного использования методов приложениями, включая распределенные объекты, серверы приложений, мониторы TP (обработки транзакций), среды разработки и просто создание нового приложения, т.е. комбинации двух или более приложений. Имеется два основных подхода: можно создать набор серверов приложений общего использования, который размещается на физически общедоступном сервере, например, на сервере приложений, или можно совместно использовать уже существующие методы внутри приложений, используя технологию распределенного доступа к методам такую, как распределенные объекты.

ИП уровня пользовательского интерфейса. В этом случае приложения связываются через их пользовательские интерфейсы как общую точку интеграции. Например, приложение, которое не предоставляет доступа на уровне базы данных или бизнес-процесса, может быть доступно через пользовательский интерфейс.

Во многих случаях это может быть единственным решением.

1.6.2Роль СУВ и постановка проблем

Типичная вычислительная инфраструктура предприятий и ведомств характеризуется фактической изолированностью составляющих ее ИС, что с точки зрения бизнеса ведет к задержкам выполнения деловых процессов, нарушению взаимодействия между подразделениями, препятствует управлению и контролю, приводя к снижению

эффективности работы организации в целом. В общем случае, отсутствие интегрированности ставит под вопрос полезность ИТ.

Технических подходов к решению задачи интеграции немало. Различные подходы можно разделить на несколько групп.

Прежде всего, это кардинальный подход, требующий полной замены существующих систем. Это позволяет заранее предусмотреть и заложить взаимосвязи между отдельными компонентами уже на этапе проектирования. Очевидно, применимость этого подхода ограничена. За достоинства новой технологии придется расплачиваться потерей уже сделанных наработок, людских и материальных инвестиций. Кроме того, принцип полной замены не решает проблем интеграции для будущих усовершенствований: новая система, как правило, не покрывает всю прикладную функциональность, которая может потребоваться предприятию, - следовательно, проблема интеграции остается.

Второй подход, связанный с переделкой механизмов доступа к старым системам при помощи новых интерфейсов, предусматривает написание различного рода обертывающих и представляющих программ. Подход этот более экономичный, поскольку позволяет сохранить существующие системы. Но в то же время он позволяет реализовывать достаточно ограниченные сценарии взаимодействия между системами.

Более глубокий интеграционный подход должен обеспечить взаимодействие систем для различных сценариев, выстраивание сложных процессов передачи данных и управления между участниками с обеспечением реакции на различные типы событий. Гибкое интеграционное решение должно быть открытым для будущих изменений.

Большинство современных ИС используют реляционные, объектные и объектно-реляционные базы данных, как основу для хранения и доступа к прикладным и системным данным. Поэтому часто под интеграцией понимают частный, но очень важный, случай интеграцию данных - обмен данными между ИС, выполнение запросов к распределенной гетерогенной совокупности ИС. Здесь основной целью интеграции ставится выделение общей модели данных, языка запросов и форматов представления данных, синхронизация источников данных, с которыми работают различные ИС. Вопросы программных интерфейсов, протоколов обмена данными отходят на второй план, а прикладная логика систем на третий. Синхронизация данных означает отсутствие взаимодействия прикладных процессов и компонентов различных систем, вместо него имеет место взаимодействие компонентов хранения данных.

Работа составного приложения подразумевает взаимодействие процессов, исполняющихся в различных ИС, взаимодействие, позволяющее приложению реагировать на события, произошедшие в других системах. Сценарий такого

взаимодействия может варьироваться от достаточно простого до весьма сложного. Для целостной реализации распределенного делового процесса, охватывающего несколько систем, уже недостаточно простой «соединительной» функциональности. Интеграционное решение должно также отслеживать сложную логику взаимодействия систем, использовать определенные исполнительные и контролирующие правила целостности. В то же время интеграционные задачи могут очень сильно различаться по своим характеристикам: временным, транзакционным, объемным. Скажем, пакетное перемещение данных, происходящее редко, налагает иные требования, чем перемещения, выполняемые в режиме близкого к реальному времени. Транзакционные перемещения и взаимодействия требуют, в свою очередь, поддержки дополнительных интеграционных механизмов.

Разнообразие условий привело к значительному разнообразию в реализациях программных решений интеграции, однако со временем выработалась типичная функциональность и архитектура.

К основным принципов проектирования программных решений интеграции обычно относят:

- Максимальное использование промышленных стандартов, стандартных продуктов
- Покомпонентная разработка - локализация функционала в независимых сервисах со стандартными интерфейсами между ними
- Масштабируемость
- Гибкость и расширяемость
- Надежность
- Высокая производительность

Среди основных компонентов реализации среды интеграции выделяют:

- транспортную инфраструктуру между ИС и приложениями;
- адаптеры и коннекторы, привязывающие прикладные системы к транспортной инфраструктуре;
- систему управления взаимодействиями (СУВ) или интеграционный сервер, берущий на себя разнообразную обработку и исполнение вычислительных процедур и процессов, связанных с интеграцией.

Транспортная среда должна обеспечивать высокую надежность передачи данных между ИС, необходимую гибкость для различных сценариев взаимодействия по сравнению со стандартными сетевыми службами, в ряде случаев, - транзакционный

режим передачи. Кроме того, от интеграционной транспортной среды требуется еще большой набор возможностей, таких как система гарантированной доставки сообщений, передача в соответствии с приоритетами, обеспечение защиты и достоверности сообщений, делегирования прав и т.п. Проблемы, решаемые транспортной средой, коннекторами, соответствующими элементами адаптеров, относят к категории **технической интероперабельности**. Это требует решения как минимум следующих задач.

Какие протоколы использовать? Протоколы разделяются по своему назначению. Выбор протоколов в рамках интеграционной системы влияет не только на потенциальную функциональность, но и на открытость системы.

Как специфицировать интерфейсы? От выбора и организации интерфейсов существенно зависит, насколько сложной окажется совместная поддержка среды интеграции существующими ИС. Реализация среды должна исходить из многоуровневой архитектуры соответствующих сервисов и служб. Сервисы верхних уровней должны решать свои задачи, основываясь на услугах сервисов нижних уровней. Как от выбора стека протоколов, так и от выбора стека интерфейсов зависит функциональность и открытость системы.

Из необходимости интеграции разнородных данных, обеспечения разных уровней интероперабельности между ИС интеграционной среды в адаптерах за гранью технических проблем взаимодействия выделяют следующие уровни требований к обмену сообщениями.

Универсальные выразительные возможности. Формат обмена должен позволять представлять любой вид данных, поскольку нельзя учесть интересы всех потенциальных систем.

Синтаксическая интероперабельность. Формат обмена должен позволять легко создать или получить синтаксические анализаторы, прикладные интерфейсы, необходимые для манипулирования данными. Приложения должны иметь возможность читать данные и получать их представление, с которыми они смогут работать.

Семантическая интероперабельность. Это наиболее важное и сложное требование к формату обмена состоит в том, что данные должны быть понятными. Это связано с установлением соответствия между терминами, используемыми в данных, что требует анализа содержимого.

Рассмотрим основную совокупность проблем, стоящих перед СУВ. Задачей СУВ является интеграция систем, обладающих характеристиками:

- Существующих систем, то есть интегрируемые системы на момент интеграции уже функционируют независимо друг от друга, обладая определенной деловой функциональностью.
- Распределенности, то есть территориального и/или логического разнесения между собой, что обуславливает необходимость использования разнородных протоколов и каналов связи при интеграции
- Разнородных (гетерогенных), то есть системы реализованы в некоторой программно-аппаратной среде, имеют свои форматы (ФД), структуры (СД) и модели данных (МД)

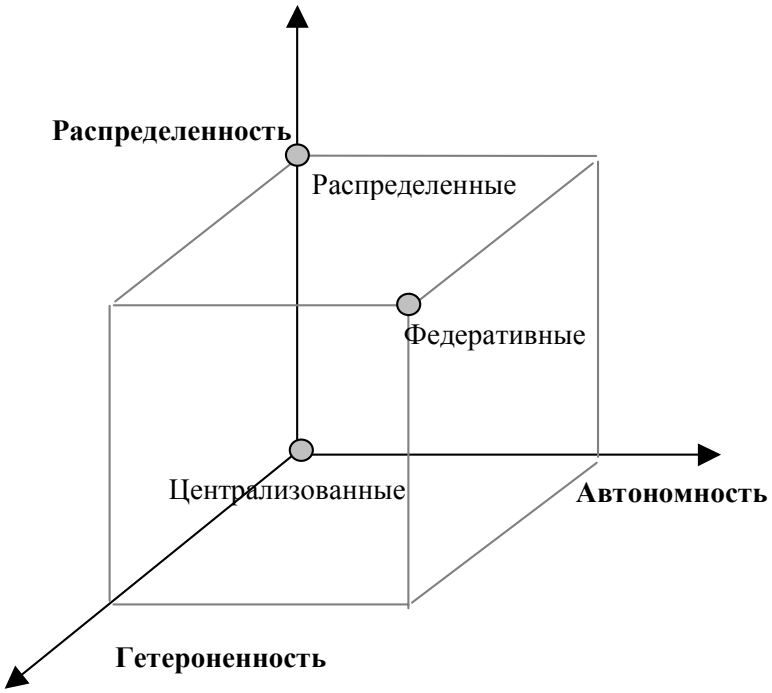


Рисунок 3. Существенные характеристики интегрируемых систем

Автономность интегрируемых ИС обуславливается автономностью:

- моделирования, проектирования данных и процессов, которое осуществляется предпочитаемым образом, используя предпочитаемую или доступную модель данных, приводит к неоднородности обработки данных, которую приходится преодолевать вспомогательными средствами (адаптерами),
- выполнения операций, которые применяются в предпочитаемом или возможном объеме функциональности, приводит к несогласованности в выполнении операций, приходится
    - компенсировать недостающую функциональность,
    - обеспечивать «совместное выполнение – распределенные транзакции;

- коммуникаций
    - в предпочитаемых или возможных способе и объеме, что ограничивает доступ к ресурсам, в частности ограничивает возможности глобальной системы в целом.

Гетерогенность ИС характеризуют следующие уровни неоднородности:

- техническая неоднородность (обмена данными, сообщениями), в первую очередь, обусловленная разнородностью «коммуникаций» (протоколов взаимодействия и соответствующих программных интерфейсов и платформ), включающей разнообразие протоколов доступа (файловый сервер, SMTP, FTP, TCP, HTTP, SOAP, Corba, DCOM …), которые могут как удерживать соединение (Tcp, Ftp, Z39.50) так и нет (HTTP), которые могут иметь разные процедуры доступа и защиты доступа.
- синтаксическая неоднородность – это различие в представлении элементов данных, характеризующееся различием:
    - в моделях данных, например,
        - реляционная, объектно-ориентированная, XML, слабоструктурированная
- разнородности представления данных, например,
    - бинарные, текстовые, XML представления данных
- функциональности методов доступа к данным, которые включает различное назначение и выразительность «языков запросов», ограничения на виды запросов. Например, на высокотехнологическом уровне - SQL, OQL, XQL или их диалекты, на низкотехнологическом уровне - обращение к файлам, вызовы функций, методов, представляемые разными API), (соединение - joins).
- семантическая неоднородность, выражающаяся в разном понимании смысла элементов данных, соответственно в разной интерпретации данных, что может быть обусловлено следующим:
    - семантика моделей данных не совпадает;
    - семантика моделей данных совпадает, но структурное представление разное;
    - представления совпадает, но механизмы расширения разные.

Ключевой и важнейшей задачей интеграции является **интеграция данных**, которая представляет собой такое «объединение» данных существующих систем, что обеспечивается:

- поддержка согласованных наборов данных
- выполнение обращений, запросов ко всем ИС сразу.

Следуя вышеуказанному, проблемы интеграции данных обуславливаются различием в:

- протоколах доступа,
- моделях данных,
- схемах данных,
- представлении (форматах и элементарных типах) данных,
- возможностях обработки данных, в частности запросах.

При создании интеграционной среды федеративных ИС, при организации доступа к ним (в частности поиска) возникают серьезные проблемы, связанные в основном со следующими обстоятельствами:

1. *Распределенность* ведет к ряду новых специфических задач, например, обеспечения связывания и интеграции независимо сопровождаемых источников информации, динамического формирования тематических коллекций, маршрутизации запросов, балансировки нагрузки.
2. Системы содержат различные коллекции информации для использования большим числом пользователей, имеют разнообразное аппаратное и программное обеспечение, причем вытекающие отсюда вычислительные и коммуникационные технические проблемы - лишь один из аспектов *проблемы интероперабельности* коллекций и их сервисов.

Интеграция информации подразумевает обеспечение следующих уровней взаимодействия между отдельными ИС:

1. **обмен данными:** ИС должна предоставлять средства, облегчающие и автоматизирующие импорт и экспорт данных, обмен данными между ИС;
2. **единообразный доступ**; ИС должна обеспечивать унифицированный механизм доступа к найденным ресурсам, вне зависимости от конкретных ИС, в которых они располагаются, и базовых протоколов доступа, используемых внутри этих репозиториев.

3. **совместные запросы**: ИС должна обеспечивать средства маршрутизации запросов, обслуживания их результатов, предоставления информации о способах доступа к найденным ресурсам;

Общепринятый подход к интеграции данных состоит в том, что выделяются:

- интегрирующая модель данных (каноническая модель данных), обеспечивающая возможность формирования одного преобразователя (адаптера канонического вида данных) для каждой ИС, а не преобразователя для каждой пары интегрируемых ИС.
- интегрирующие схемы данных (канонические схемы, подсхемы данных, представления) интегрирующей схемы данных, обеспечивающие согласованный вид всей совокупности информации интегрируемых ИС для отражения интересов как «пользователей» интегрированной среды, так и самих интегрируемых ИС.
- схемы данных экспорта интегрируемых ИС, в которых унифицируется исходные модель и схемы данных интегрируемой ИС, обобщаются функции доступа к ней

Соответствующая концептуальная модель интеграции информации имеет вид

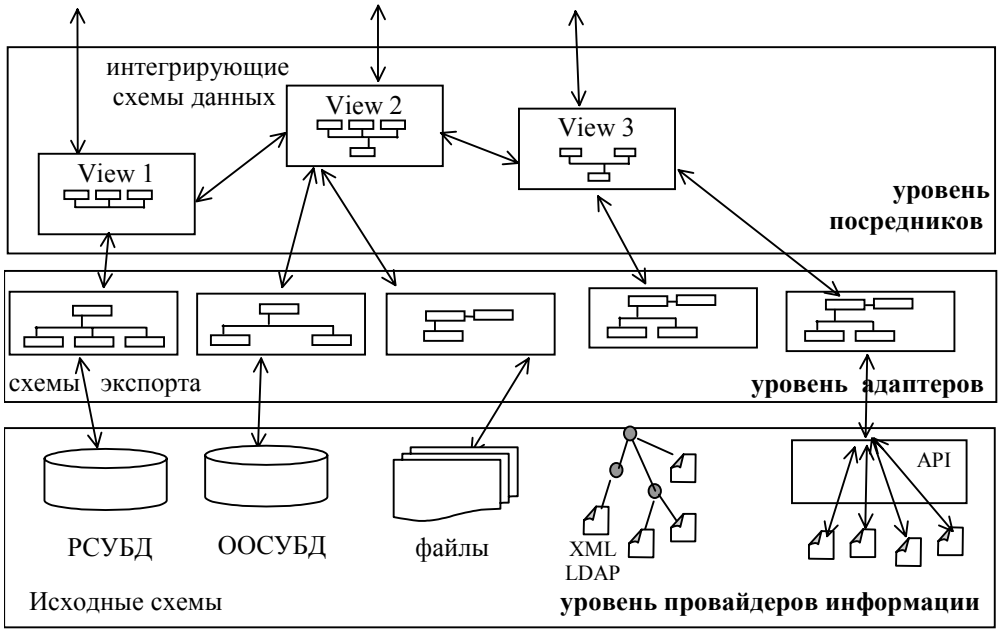


Рисунок 4. Концептуальная модель интеграции информации

Ее архитектурное выражение, например, может иметь следующий вид

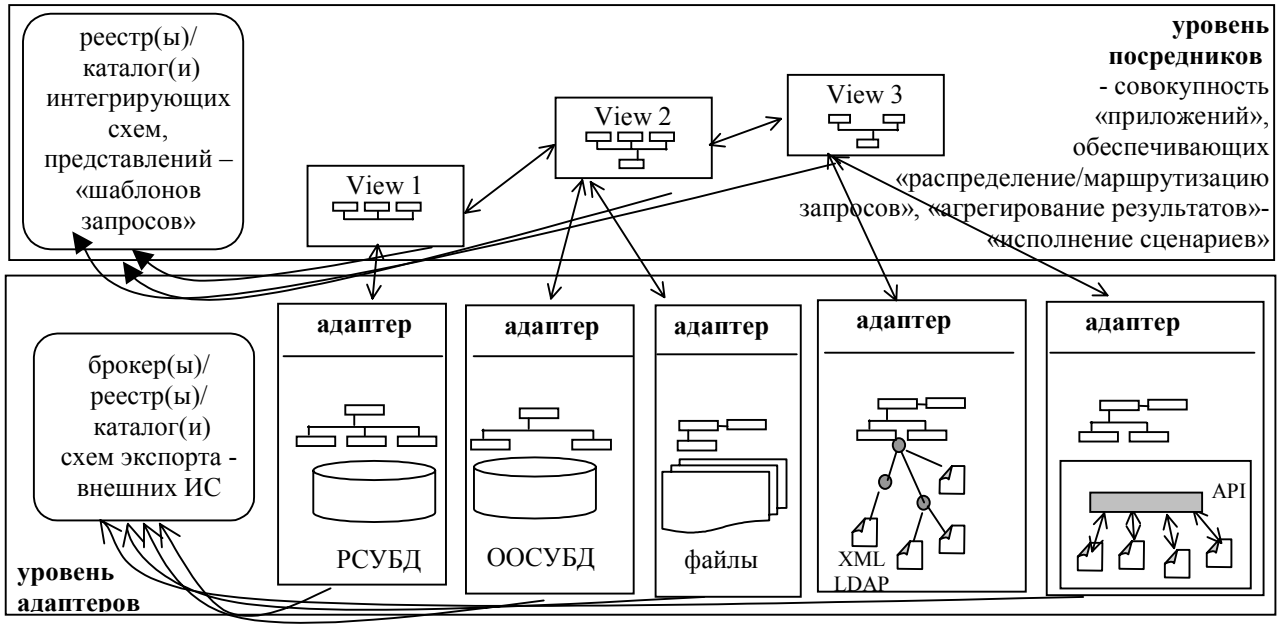


Рисунок 5. Пример архитектурного выражения концептуальной модели интеграции информации

Ясно, что аналогичное архитектурное выражение имеет место как на более высоком уровне – уровне деловой логики, так и на более низких технических программных уровнях, где

- «схемы экспорта» представляются «адаптерами», их интерфейсами (например, Web-сервисами, в низкоуровневой реализации - «каталогами файловой системы», «SMTP-серверами», куда помещаются «файлы-запросы» и откуда извлекаются «файлы-ответы»),
- «интегрирующие схемы» - «посредниками» (СУВ), их сценариями (например, декларациями рабочих процессов ПЭВ, ЭАР)

Можно выделить следующие базовые разрезы, для каждого из которых должна быть разработана аналогичная модель:

- метаданные, семантические схемы данных;
- контролируемые таксономии (словари, классификаторы, тезаурусы терминов);
- глобальная идентификация общих ресурсов;
- кооперация сервисов (обмен сообщениями, координация, обеспечение транзакций действий);
- журнализация и аудит выполняемых действий;
- описания рабочих процессов, интегрирующих сервисы;

- аутентификация пользователей, сервисов и услуг, обеспечения единого входа (single sign on), авторизация и делегирования доступа
- регистрация, контроль, сертификация и использование внесистемных сервисов и услуг
- организационные процедуры поддержки стандартов.

2 Общие вопросы интеграции

При построении модели взаимодействия ИС посредством СУВ можно выделить следующие архитектурные уровни:

1. **Уровень адаптеров.** Каждая интегрируемая информационная система обязана представлять свой внешний интерфейс некоторым унифицированным образом. Другими словами, она должна определять *сервис* с определенным набором *операций*. Введение понятия сервисов ИС сразу выявляет ряд базовых вопросов, которые должны быть решены для возможности взаимодействия ИС в рамках СУВ:
    - Выбор *модели описания интерфейсов сервисов ИС* и общей архитектуры их построения.
    - Выбор *протоколов взаимодействия* и *транспортной среды* для сообщений, передаваемых между сервисами ИС. При этом выбранные протоколы общения с адаптерами ИС должны обеспечивать:
        - *целостность и гарантированную доставку* передаваемых данных;
        - *безопасность передачи данных*, что становится особенно актуальным при использовании в качестве транспортной среды сети Интернет;
2. **Уровень посредников.** На данном уровне определяются регламенты взаимодействия сервисов ИС (рабочие процессы), предназначенные для выполнения общей задачи, поставленной перед СУВ. Для реализации механизмов данного уровня необходимо решить следующие задачи:
    - *Выбор единой унифицированной модели данных*, в рамках которой происходит взаимодействие ИС и интеграция передаваемых данных (определяющей структуры данных, передаваемые в сообщениях между сервисами ИС). При этом для возможности интеграции данных необходим механизм *глобальной идентификации* всех информационных ресурсов взаимодействующих ИС.
    - Необходимо идентифицировать имеющиеся в распоряжении СУВ информационные системы. Для этого необходимо разработать *модель регистрации сервисов ИС* с возможностью динамического поиска необходимого участника взаимодействия. Для построения динамической и гибкой модели взаимодействия ИС необходимо предусмотреть

возможность регистрации дополнительной метаинформации по участвующим в общем процессе ИС.

- После того, как определен набор участников процесса, необходимо формализовать регламент (протокол верхнего уровня) их взаимодействия. Для этого надо определить модель описания управляющего потока и потока данных в рамках данного регламента – выбрать *средство описания автоматизированных рабочих процессов*. При этом для возможности согласованного выполнения поставленной задачи СУВ должна обеспечивать поддержку координирующих протоколов, позволяющих гарантировать целостность выполняемых в рамках процесса операций, что выливается в поддержку некоторой *модели транзакционности*.

Ввиду этого анализ основных интеграционных механизмов в этом и последующих разделах проводится в разрезе следующих аспектов интеграции, которые полностью определяют архитектуру интегрированной среды взаимодействия ИС, описанную выше:

- информационные системы и сервисы;
- транспорт и сообщения;
- каноническая модель данных;
- регистрация сервисов и метаинформации;
- безопасность;
- транзакции;
- рабочие процессы;
- глобальная идентификация.

В данном разделе приводится обзор основных механизмов, используемых при интеграции распределенных информационных систем посредством СУВ, в разрезе указанных аспектов интеграции. По каждой группе механизмов приводится анализ и сравнение существующих подходов к решению поставленной проблемы, на основе которого делается вывод и рекомендации к применению данных механизмов.

2.1 ИС и сервисы

Web-сервисы: сервис-ориентированная архитектура

В последние годы знаменуются активностью, сконцентрированной вокруг разработки Web-сервисов. Одна из ключевых причин интереса к Web-сервисам заключается в том, что Web-сервисы хорошо подходят для реализации сервис-ориентированной архитектуры (COA). При использовании Web-сервисов для построения сервис-ориентированной архитектуры решения заключаются в коллекциях автономных сервисов, идентифицируемых URL, с интерфейсами, документированными WSDL, и обработки правильно определенных XML сообщений. COA – естественное дополнение к объектно-ориентированному (OO), процедурному и основанному на данных подходов к реализации решения. На самом деле при создании COA системы, отдельные сервисы обычно строятся с использованием одной или более из этих технологий.

Сервис-ориентированная архитектура отличается от OO и процедурных систем в одном ключевом аспекте: технологии *связывания*. Взаимодействие сервисов основывается на том, *какие* они предоставляют функции и *как* они их предоставляют. OO и процедурные системы связывают элементы друг с другом на основании типов или имен. В отличие от предыдущих систем, модель Web-сервисов не оперирует с понятием разделяемых типов, требующих общей реализации. Вместо этого сервисы взаимодействуют на основе описаний (WSDL/BPEL4WS для обработки сообщений) и схем (WSDL и XSD для структуры сообщений). Это позволяет сервису описать структуру сообщений, которые он может посылать и/или получать. Разделение между структурой и поведением и явное машинно-проверяемое описание этих характеристик упрощает интеграцию в разнородные среды.

Кроме того, эта информация достаточно характеризует интерфейс сервиса, так что для создания структуры сообщений или поведения интеграция приложения не требует общей среды исполнения.

Сервис-ориентированная модель предполагает полностью распределенную среду, что делает ненужным распространение изменений всем участникам, использующим сервис. По этой причине технологии, разработанные для использования в сервис-ориентированном проектировании, обеспечивают больше гибкости, чем традиционные объектно-ориентированные интерфейсы. Процедурное и объектно-ориентированное проектирование обычно приравнивает совместимость типов с семантической совместимостью. Сервис ориентация предоставляет более богатую модель для определения совместимости. Структурная совместимость основывается на описании взаимодействия (WSDL и возможно BPEL4WS) и схеме (XSD) и может быть проверена. Предыдущие подходы к распределенным приложениям явно предполагали общее пространство типов, модель исполнения и процедурную/объектную модель ссылок. По

существу, модель распределенной системы определялась моделью программирования «в памяти». Сервис ориентация просто предполагает, что сервисы исполняются автономно и не существует понятия локального исполнения или общей операционной среды. По этой причине COA просто предполагает, что ошибки коммуникаций, доступности и типов обычны. Кроме того, в отличие от модели «в памяти» сервис-ориентация предполагает, что входящее сообщение может не только быть повреждено, но и что оно может быть послано с целью навредить или вообще с непредсказуемой целью.

Развернув платформу на базе сервис-ориентированной архитектуры (COA), организация получает широкие возможности при определении того, какие из своих служб можно сделать доступными для клиентов, партнеров и поставщиков. С другой стороны, клиенты, партнеры и поставщики получает в свое распоряжение более обширный набор предлагаемых сервисов. Технологии Web-сервисов повысят продуктивность и помогут компаниям более эффективно реагировать на изменяющиеся требования рынка. В отличие от других технологий электронного бизнеса, архитектура Web Services Architecture (WSA) обеспечивает полную независимость от платформы реализации посредством определения набора интегрирующих Web-стандартах, в рамках которых осуществляется взаимодействие, облегчая тем самым процесс межсистемной интеграции и уменьшая трудности взаимодействия для пользователей. Web-сервисы потенциально могут обеспечить и более высокую прибыль на инвестированный капитал за счет сокращения сроков и затрат на внедрение новых приложений, широкого внедрения Web-технологий и роста доходов.

В частности, целью технологии Web-сервисов является ликвидация многих ограничений, накладываемых современными IT-технологиями на взаимодействие организаций, и, как следствие этого, повышение качества бизнес-процессов, что в итоге позволяет достичь лучших конечных результатов. Поставщики технологий Web-сервисов в состоянии предложить широкий ассортимент продукции и обладающие интеграционными возможностями программные и аппаратные средства, сервисы, а также партнерские услуги, поддерживающие эту новую многообещающую технологию.

2.2 Транспорт и сообщения

2.2.1 Постановка задачи

Протокол обмена сообщениями между средами предназначен для решения следующих задач:

- Осуществление операций, требующих совместной работы нескольких различных информационных систем. К числу таких операций в частности относится агрегирование информации о ресурсе, данные о котором хранятся в нескольких различных системах, или операции, в результате которых изменяется состояние нескольких систем;
- Обеспечение доступа информационных систем к глобальным ресурсам среды (например, к реестру сервисов).

Протокол должен удовлетворять следующим требованиям:

- Поддержка различных операционных систем и транспортных протоколов;
- Расширяемость: возможность адаптировать протокол к задачам среды.
- Должна быть возможность обработки ошибочных ситуаций с доступом к информации об ошибке и месте её возникновения (транспортный уровень, внутренняя ошибка вызываемой стороны);
- Поддержка различных типов данных, передаваемых в сообщениях, в том числе возможность передачи двоичных файлов;
- Возможность применения для выполнения транзакций.

2.2.2Обзор и анализ

В данном разделе рассмотрены популярные технологии, использующиеся для обеспечения взаимодействия в распределённых средах. Протоколы рассмотрены в хронологическом порядке их становления.

2.2.2.1CORBA

CORBA (Common Object Request Broker Architecture) – открытая, независимая архитектура и инфраструктура, которая применяется для реализации приложений, работающих совместно в сетевых окружениях. Используя стандартный протокол IIOP, приложение, построенное на основе CORBA, разработанное любым производителем, на любом компьютере и для любой операционной системы, написанное на любом языке программирования может взаимодействовать с CORBA – приложением того же или другого производителя, на практически любом другом компьютере, операционной системе и языке программирования.

Приложения CORBA состоят из объектов – индивидуальных фрагментов работающего приложения, которые содержат код и данные, которые часто (но не всегда) представляют понятие реального мира. Обычно существует множество экземпляров объектов одного и того же типа – например, приложение продаж через Интернет может

содержать множество объектов типа “корзина”, которые идентичны по функциональности, но отличаются тем, что приписаны различным покупателям, и содержат информацию о товарах, выбранных данным покупателем. Для некоторых типов может быть только один экземпляр объекта. Например, когда функциональность существующего приложения требуется сделать доступной в среде CORBA, нужно представить его в виде объекта, у которого будет один экземпляр.

Для каждого типа объектов определяется интерфейс на языке OMG IDL. Этот интерфейс определяет синтаксис, в соответствии с которым к нему могут обращаться клиенты. Любой клиент, который будет взаимодействовать с объектом, должен использовать этот интерфейс для специфицирования вызываемых методов и их аргументов.

2.2.2.2XML-RPC

XML-RPC – протокол удалённого вызова методов, работающий посредством Internet. Цели создания XML-RPC:

- Обход сетевых фильтров. Протокол использует только возможности CGI;
- Понятность. Формат сообщений должен быть простым, насколько это возможно. По внешнему виду документа с XML-RPC сообщением должно быть понятно, что оно делает;
- Простота реализации. Реализация протокола должна быть простой, в связи с чем он может быть легко адаптирован для работы в различных средах и операционных системах.

Сообщение XML-RPC представляет собой HTTP-POST запрос. Содержанием запроса является XML документ. Метод выполняется на сервере, и значение, которое он возвращает, так же оформляется в виде XML. Параметрами процедуры могут быть скалярные значения, числа, строчки, даты, и так далее. Так же параметры могут являться сложными структурами или списками.

Ниже приведён пример XML-RPC запроса:

```
POST /RPC2 HTTP/1.0
User-Agent: Frontier/5.1.2 (WinNT)
Host: betty.userland.com
Content-Type: text/xml
Content-length: 181
```

```
<?xml version="1.0"?>
<methodCall>
  <methodName>examples.getStateName</methodName>
  <params>
        <param> <value><i4>41</i4></value> </param>
  </params>
</methodCall>
```

Как видно, с сообщением передаётся XML документ с корневым тэгом <methodCall>. Этот тег должен содержать элемент <methodName> - строчку, содержащую имя метода, который следует вызвать. Строчка может содержать только символы, допустимые в идентификаторах: символы алфавита, цифры, точки, а так же “/” и “-”. Интерпретация данной строчки возлагается на сервер.

Пример ответа на XML-RPC запрос:

```
HTTP/1.1 200 OK
Connection: close
Content-Length: 158
Content-Type: text/xml
Date: Fri, 17 Jul 1998 19:55:08 GMT
Server: UserLand Frontier/5.1.2-WinNT
<?xml version="1.0"?>
<methodResponse>
  <params>
        <param> <value><string>South Dakota</string></value> </param>
  </params>
</methodResponse>
```

2.2.2.3SOAP

SOAP является легковесным протоколом обмена информацией в децентрализованной, распределённой среде. Он использует технологии XML для специфицирования основы расширяемого протокола и определяет форму сообщения, которая может быть передана по различным транспортным протоколам.

SOAP состоит из трёх частей:

- “Конверт” SOAP определяет в общем как должно выглядеть сообщение, кто должен его обрабатывать и что является обязательным или необязательным;

- Правила кодирования SOAP определяют механизм записи, который может быть использован для обмена пользовательскими типами данных;
- Соглашению по использованию SOAP как протокола для удалённого вызова методов.

Сообщения SOAP представляют собой одностороннюю передачу данных от источника к получателю. В то же время их можно комбинировать для реализации шаблонов взаимодействия, таких как запрос/ответ.

Реализации SOAP могут быть оптимизированы для функционирования в специфическом сетевом окружении. Например, привязка SOAP сообщений к протоколу HTTP позволяет переслать SOAP ответ как ответ на HTTP запрос, с использованием того же соединения, по которому было передано исходное сообщение. Независимо от протокола, поверх которого работает SOAP, сообщения передаются вдоль так называемого “пути”, который позволяет обрабатывать сообщения на одном или нескольких промежуточных узлах, в дополнении к точке назначения.

Приложения SOAP при получении сообщения должны его обработать в соответствии со следующими правилами:

1. Идентифицировать все части сообщения SOAP, предназначенные для данного приложения.
2. Проверить что все обязательные для обработки части, выявленные на первом шаге, поддерживаются данным приложением и обработать их в соответствии с правилами. Этот процесс может проигнорировать необязательные для обработки части.
3. Если данное приложение не является конечным адресатом сообщения, то следует удалить все обработанные на втором шаге части перед пересылкой сообщения дальше.

Примеры SOAP сообщений (запрос/ответ)

```
<SOAP-ENV:Envelope
 xmlns:SOAP-ENV="http://schemas.xmlsoap.org/soap/envelope/"
 SOAP-ENV:encodingStyle="http://schemas.xmlsoap.org/soap/encoding/">
  <SOAP-ENV:Body>
     <m:GetLastTradePrice xmlns:m="Some-URI">
        <symbol>DIS</symbol>
```

```
      </m:GetLastTradePrice>
    </SOAP-ENV:Body>
  </SOAP-ENV:Envelope>

  <SOAP-ENV:Envelope
    xmlns:SOAP-ENV="http://schemas.xmlsoap.org/soap/envelope/"
    SOAP-ENV:encodingStyle="http://schemas.xmlsoap.org/soap/encoding/"/>
     <SOAP-ENV:Body>
       <m:GetLastTradePriceResponse xmlns:m="Some-URI">
          <Price>34.5</Price>
       </m:GetLastTradePriceResponse>
     </SOAP-ENV:Body>
  </SOAP-ENV:Envelope>
```

При разработке протокола SOAP основное внимание было уделено простоте и расширяемости. Для достижения этого из базовой спецификации исключены возможности, которые часто встречаются в распределённых средах. В частности, это надёжность, безопасность, маршрутизация и шаблоны обмена сообщениями. Данные возможности специфицируются расширениями протокола.

Таблица 1. Расширения базового протокола SOAP.

Название стандарта	**Назначение**
WS-Reliable Messaging	Обеспечивает надёжную доставку сообщений между распределёнными приложениями с учётом возможности сбоев в программных компонентах, операционных системах и каналах связи
WS-Coordination	Специфицирует основу для создания расширений, которые согласовывают действия распределённых приложений
WS-Transaction	Предоставляет возможность получения согласованного результата действий нескольких приложений (основана на WS-Coordination)
WS-Security	Определяет механизмы обеспечения

	целостности сообщений, конфиденциальности, и аутентификации
WS-Addressing	Предоставляет независимый от транспортного протокола механизм указания адреса сервисов
WS-Notification	Рассылка сообщений в соответствии с моделью “публикация/подписка”
WS-Acknowledgement	Поддерживает надёжный обмен сообщениями, гарантируя доставку “в точности один раз”, или “по крайней мере один раз”
WS-CallBack	Указывает адрес для асинхронного ответа на сообщение
WS-Federation	Определяет механизмы работы с идентифицирующей пользователя информацией в распределенной среде
SOAP Messages With Attachments	Позволяет пересылать вместе с сообщениями произвольные данные, такие как графические изображения или документы

Спецификация SOAP with Attachments описывает абстрактную составную структуру, состоящую из главной части, содержащей SOAP-сообщение, и связанных с ним вторичных частей – вложений, в которые, собственно, и помещаются бинарные данные. Каждая часть такой структуры характеризуется одним или более URI-указателем, используемым для ссылок на него из других частей. Главной и вторичной частям присвоены имена SOAPMessage и SecondaryPartBag относительно некоего базового URI.

Составная структура не является ни обобщением структурной модели SOAP, ни обобщением конверта SOAP и не определяет модель обработки основного сообщения (все эти задачи отводятся спецификации SOAP). Это просто абстрактная модель, которой нужно следовать при дальнейшей реализации привязок SOAP к конкретному транспортному протоколу. В действительности спецификация склоняется к привязкам к протоколу HTTP.

Вот примеры возможного применения SOAP Attachment Feature:

- главная часть и бинарные данные могут быть инкапсулированы в одно MIME-сообщение (см. WS-Attachments) и переданы через TCP или HTTP;
- главная часть и бинарные данные могут быть инкапсулированы в MIME Multipart/Related message и переданы через HTTP;

- главная часть может быть послана через HTTP без инкапсуляции, а бинарные данные получены по отдельному требованию приложения командой HTTP GET.

Спецификация также постулирует некоторые требования, вытекающие из вопросов безопасности данных во вложениях. Обработка вторичных частей определяется не ею, а семантическими конструкциями SOAP, специфическими для тех программ, для которых вложения предназначены.

2.2.2.4Системы сообщений

Система сообщений – распределённая система, основанная на асинхронном обмене сообщениями между компонентами. Приложения, использующие такие системы, не взаимодействуют друг с другом непосредственно (по сравнению, например, с RMI). Если один компонент системы хочет послать сообщение другому компоненту, он посылает сообщение системе сообщений, и после этого система доставляет сообщение адресату.

Такой подход обладает следующими особенностями:

- Приложение, которое послало сообщение, не обязано ожидать ответа, и может продолжать свою деятельность.
- Ни приложение, которое отсылает сообщение, ни принимающая сторона не должны быть активны в какой-либо момент времени. Если адресат сообщения неактивен, сервис гарантирует, что сообщение будет доставлено сразу, как адресат станет активным.
- Компоненты системы не соединены непосредственно друг с другом. Это позволяет, в частности, переносить компоненты на другие серверы без нарушения работы системы.

Существуют две базовые модели обмена сообщений: модели пересылки сообщения от одного узла к другому, и модель публикации/подписки. Первая модель применяется в том случае, когда одному из компонент необходимо переслать сообщение в точности одному компоненту (получателю). Данная модель основана на понятии очереди сообщений: отправитель помещает сообщение в очередь, а получатель читает сообщение из очереди. При этом могут существовать несколько получателей, читающие из одной и той же очереди (сообщение будет доставлено только одному из них).

Модель публикации/подписки применима, одному или нескольким компонентам (отправителям) необходимо послать сообщение, предназначенное для одного или нескольких компонент (получателям). Метод основан на понятии “тематики” сообщений –

отправители посылают сообщение для данной темы, а получатели читают данные сообщения.

JMS - Java API для работы с системами сообщений. С помощью JMS приложение на Java может создавать, посылать, принимать и читать сообщения. JMS может работать с множеством доступных продуктов (iPlanet Message Queue, IBM MQSeries, Progress Software SonicMQ, BEA WebLogic Server, Prism Technologies OpenFusion)

2.2.3Выводы

Рассмотренные протоколы взаимодействия систем удовлетворяют предъявленным функциональным требованиям. Поэтому выбор протокола следует осуществлять на основании таких характеристик, как открытость, число и качество доступных реализаций, состав и количество организаций, поддерживающих и развивающих стандарт.

Недостатком существующих в настоящее время систем сообщений является то, что они основаны на собственных решениях разработавших их организаций. Например, не существует общего стандарта, регламентирующего формат пересылки сообщений. Хотя такие средства как JMS позволяют избежать зависимости от выбранной реализации системы сообщений в программном коде, такая зависимость неизбежно проявится на уровне интеграции приложений.

Недостатком архитектуры CORBA в рассматриваемом контексте является её ориентированность на удалённый вызов методов, и, как следствие, неудобство в использовании для передачи информационных сообщений. Например, сложно реализовать сценарий обработки запроса с использованием электронной почты в качестве транспорта, в то время как с использованием SOAP это осуществляется сравнительно просто. CORBA успешно применяется для распределённых вычислений внутри организаций, но в среде Интернет возможны проблемы из-за наличия сетевых фильтров (настроенных, как правило, на HTTP трафик), а так же ненадёжности или невысокой пропускной способности каналов связи.

В результате, указанным характеристикам лучше всего соответствует протокол SOAP, который в настоящий момент используется как основной протокол взаимодействия информационных систем наиболее влиятельными компаниями в области информационных технологий. В частности, поддержка этого протокола включена в популярные платформы для разработки web приложений – Java (Sun) и .NET (Microsoft).

Следует отметить то, что в случае SOAP часть требований не выполняются в рамках базовой спецификации, но могут быть включены протокол – одной из главных характеристик SOAP является его расширяемость. Данные расширения могут быть

сделаны либо на основе существующих стандартов, либо, при необходимости, непосредственно на основе требований СУВ. Это является ещё одним неоспоримым преимуществом протокола, поскольку, в случае использования других решений, пришлось бы следовать интегрированными с ними решениями (такими, как система безопасности), которые могут не соответствовать предъявляемым в рамках СУВ требованиям.

2.3 Каноническая модель данных и язык спецификации схем данных

В контексте интеграции приложений приходится иметь дело с потребностью в обмене данными между этими приложениями так, чтобы они могли понять друг друга – проблемой интероперабельности.

Интероперабельность данных включает как синтаксический уровень (где существует поверхностное однообразие подходов к формату записи информации, но практически полное доверие к человеческому интеллекту в процессе достижения согласованности), так и более глубокий уровень, на котором различные компьютерные системы разделяют понимание информации – семантическую интероперабельность.

Исторически существовало много подходов к представлению данных (они делятся на несколько основных семейств, такие как реляционная, объектно-ориентированная и иерархическая модели данных). Разные приложения, подлежащие интеграции, используют разные подходы.

Для интеграции информации, соответствующей различным моделям данных, необходима каноническая модель данных, которая была бы максимально удобна для решения рассматриваемых задач.

Под *канонической моделью данных* понимается единый способ представления данных, которому следуют все интегрируемые системы при внешнем представлении собственных данных для обмена с друг с другом. При этом каждая система внутренне представляет данные согласно собственной модели данных, а каноническая модель данных служит «общим знаменателем», позволяющим системам понять друг друга, несмотря на разные взгляды на представление информации.

Выбор канонической модели данных рассматривается далее в разделе 4.3 «Каноническая модель данных и язык спецификации схем данных» главы 4 «Обоснование обеспечения интеграции с применением Web-сервисов (Архитектуры WSA)», поскольку проблемы синтаксической и семантической интероперабельности

резонно рассматривать после выбора механизма технической интероперабельности (а именно WSA) и с учётом предпочтительного формата обмена этого механизма (а именно XML).

2.4 Регистрация сервисов и метаинформации

2.4.1 Постановка задачи

Общий реестр сервисов необходим для динамического связывания компонент распределённой системы, а также может использоваться конечными пользователями в процессе использования системы.

От реестра требуется следующая функциональность:

- Должна быть обеспечена возможность упорядочивания информации о сервисах по различным признакам, в частности, желательна поддержка классификаторов.
- Возможность поиска сервисов, зарегистрированных в реестре, по различным критериям (название, поставщик сервиса, принадлежность рубрикам какого-либо классификатора).
- Реестр должен обеспечивать возможность хранения технической информации, достаточной для вызова сервисов.

2.4.2 Обзор и анализ

Как правило, технологии, на которых построена распределённая среда, предопределяют выбор реестра сервисов. Например, при использовании технологии CORBA, для этой роли применяется сервис именования (Naming Service). Данный сервис позволяет ассоциировать абстрактные имена с объектами CORBA.

Для регистрации сервисов, доступных по протоколу SOAP, применяются стандарты UDDI и ebXML.

2.4.2.1 UDDI

В центре внимания стандарта UDDI – спецификация набора требований, поддерживающих описание и поиск:

- организаций, компаний и других поставщиков Web сервисов;
- Web-сервисов, предложенных этими организациями;
- технических интерфейсов, которые могут быть использованы для доступа к данным сервисам.

UDDI основан на общем наборе индустриальных стандартов, таких как HTTP, XML, XML Schema и SOAP. UDDI предоставляет инфраструктуру для основанных на Web-

сервисах программных средах, как для публично доступных сервисов, так и для сервисов, разработанных для функционирования в локальных сетях компаний.

Основной целью UDDI реестра является представление данных и метаданных о Web-сервисах. Реестр UDDI, независимо от области применения (внутри организации или как публично доступный сервис), предоставляет стандартные механизмы для классификации, каталогизации и управления Web-сервисами, благодаря чему они могут быть найдены и использованы. Коммерческие организации и другие провайдеры сервисов могут использовать UDDI реестр для представления информации о своих сервисах, независимо от типа использования сервиса (электронная коммерция или другие цели). UDDI реестр может быть применён, в частности, для осуществления следующих сценариев:

- найти реализации сервисов, которые основаны на общем абстрактном определении интерфейса;
- найти поставщиков сервисов, которые классифицированы в соответствии с известным классификатором или системой идентификации;
- определить уровень безопасности и транспортные протоколы, поддерживаемые данным сервисом;
- найти сервисы по ключевому слову;
- запомнить техническую информацию о сервисе и обновлять эту информацию в процессе работы приложения.

2.4.2.2ebXML

Реестр ebXML служит как центральный репозиторий, с помощью которого компании могут совместно работать с информацией. Его можно определить как набор сервисов, которые позволяют обмениваться информацией между заинтересованными сторонами с целью интеграции бизнес-процессов. Таким образом, вместе с тем, что ebXML является каталогом информации, он так же является механизмом хранения. Это место, в котором можно находить, размещать и получать информацию с целью осуществления бизнес-взаимодействий.

Информационная модель реестра ebXML определяет следующие основные классы:

- **RegistryObject** – абстрактный базовый класс для большинства классов модели. Класс содержит минимальный набор метаданных, а так же методы доступа к связанным объектам, которые предоставляют дополнительные метаданные для объекта, в частности:
    - принадлежность определённой рубрике классификатора;

- идентификаторы;
- внешние ссылки.

С экземплярами данного класса так же может быть проассоциировано произвольное количество динамических атрибутов.

- **RepositoryItem** – представляет собой объект, расположенный в репозитории, такой, как XML документ или DTD. Ассоциирован с **RegistryObject**, который содержит метаданные об объекте.
- **User** – содержит информацию о зарегистрированном в системе пользователе. В частности, используется для аудита событий.
- **Organization** – информация об организации. Может содержать ссылку на родительскую организацию. С пользователями и организациями могут быть ассоциированы почтовые адреса, адреса электронной почты, и телефоны.

По сравнению с UDDI-реестром ebXML обладает следующими возможностями:

- Хранение произвольных данных
- Поддержка жизненного цикла хранимых элементов
- Автоматическая проверка содержимого перед публикацией (в зависимости от типа информации)
- Автоматическая каталогизация содержимого
- Контроль версий
- Хранение элементов таксономий (в UDDI только пространство имен)
- Ассоциации, определяемые пользователем
- Поддержка группировки элементов
- Политики доступа, основанные на группах, ролях
- Поддержка ссылок на объекты другого ebXML репозитория, распределённые запросы
- Поддержка доступа по HTML.

2.4.3Выводы

Рассмотренные стандарты (UDDI и ebXML) соответствуют предъявленным требованиям, но задаче регистрации сервисов лучше соответствует UDDI, поскольку ebXML Registry разработан как репозиторий произвольной метаинформации. В пользу зрелости стандарта UDDI говорит так же включение в профиль WS-I, и существование глобального UDDI реестра, поддерживаемого IBM и Microsoft.

Что касается ebXML реестра, в настоящий момент непонятно, насколько широкое применение найдёт эта инициатива в индустрии. Однако имеет смысл рассмотреть использование ebXML реестра в качестве дополнения к UDDI, для хранения такой информации как, например, XSD схемы, используемые сервисами.

2.5 Безопасность

2.5.1 Постановка задачи

Основные термины и определения, связанные с безопасностью компьютерных систем, приводятся в Глоссарии по безопасности Интернета (Internet Security Glossary) [RFC2828]. Далее рассматриваются термины, относящиеся к теме настоящего документа.

Безопасность (security) – (1) меры, предпринимаемые для защиты системы; (2) состояние системы, обеспечиваемое мерами, предпринимаемыми для защиты системы; (3) состояние системных ресурсов, при котором они не подвержены несанкционированному доступу, а также умышленному или случайному изменению, уничтожению или потере.

Защита системы осуществляется посредством набора *сервисов безопасности*. Сервис безопасности – это вычислительный или коммуникационный сервис, предоставляемый системой для определенного вида защиты системных ресурсов. Примерами сервисов безопасности служат: сервис контроля доступа, сервис аудита (журналирования), сервис конфиденциальности данных, сервис целостности данных, сервис аутентификации удаленной системы, сервис аутентификации источника данных.

Разработка безопасной системы ведется на нескольких уровнях. Каждый уровень представляет отдельное видение безопасности, отвечающее на ряд вопросов от «Какие необходимы сервисы безопасности?» до «Как эти сервисы должны быть реализованы?»

На верхнем уровне определяется *политика безопасности* – набор правил и методик, которые специфицируют как система или организация обеспечивает сервисы безопасности для защиты конфиденциальных или жизненно важных ресурсов. Примерами политики безопасности могут служить: политика безопасности, основанная на идентификаторах и/или атрибутах пользователей, групп пользователей, сущностей, действующих от их имени и ресурсов, к которым осуществляется доступ; политика безопасности, основанная на правилах, предполагающая сравнение некоторых глобальных атрибутов пользователей и объектов доступа.

Модель безопасности – это формальное описание множества сущностей и связей между ними, посредством которых система реализует заданный набор сервисов

безопасности. Модель определяет такие понятия объекта, субъекта (пользователя), привилегия, уровень доступа. Наиболее широко известны модели Белла-ЛаПадулы и Харрисона-Рузо-Ульмана.

На основе модели безопасности разрабатывается *архитектура безопасности*, которая определяет:

- сервисы безопасности, которые должна обеспечивать система;
- необходимые для этого компоненты системы;
- необходимый уровень производительности этих компонент для обеспечения безопасности системы в заданной среде.

При реализации сервисов безопасности используются механизмы безопасности. Примерами таких механизмов служат: механизм обмена аутентификационной информацией, контрольная сумма, хеширование, цифровая подпись, шифрование, зашумление потока информации, назначение идентификаторов субъектам.

В настоящем разделе рассматриваются сервисы безопасности в контексте распределенных систем.

2.5.2Обзор и анализ

Рассмотрим несколько примеров, в которых возникает потребность в использовании сервисов безопасности при взаимодействии потребителей и поставщиков государственных услуг.

Сценарий 1 – Взаимодействие с одним подразделением

Физическое или юридическое лицо желает получить услугу некоторого подразделения государственного учреждения. Оно обращается к «входной странице службы» через Интернет. Там оно заполняет электронную форму, подписывает ее своей цифровой подписью в подтверждение своей личности, прикрепляет документы и сертификаты, необходимые для выполнения запроса и отсылает форму. Подразделение обрабатывает форму и выдает подтверждение с датой и подписанной квитанцией вместе с результатами выполнения запроса (последние могут быть доставлены потребителю позднее). Примерами могут служить подача заявлений на выдачу лицензии, пособия, документов и т.п.

Сценарий 2 – Изменение статуса гражданина или юридического лица

У гражданина изменились персональные данные, которые требуются для предоставления в государственные учреждения. Он имеет возможность обратиться к службе, которая позволяет получить детальную информацию о произошедших изменениях, после того, как личность гражданина установлена. Полученные данные передаются заинтересованным государственным учреждениям. Гражданин получает уведомления о прохождении информации. Примерами могут служить: изменение семейного положения, имени, фамилии, адреса гражданина.

Сценарий 3 – Электронное голосование

Гражданин желает участвовать в выборах в электронной форме. Он подтверждает свою личность, после чего ему предоставляется электронный бюллетень для голосования согласно имеющейся информации о его месте жительства. Результаты голосования собираются, подсчитываются и объединяются с результатами ручного подсчета.

В процессах взаимодействия государственных учреждений и потребителей их услуг можно выделить следующие отдельные виды взаимодействий:

- взаимодействие между физическим или юридическим лицом и службой, предоставляемой государственным учреждением;
- взаимодействие между государственными структурами или подразделениями внутри одной государственной организации;
- взаимодействие между государственными организациями и сторонними поставщиками услуг.

В рамках этих взаимодействий от системы необходима поддержка следующих сервисов безопасности: сервис аутентификации удаленной системы, сервис аутентификации источника данных, сервис контроля доступа, сервис аудита (журналирования), сервис конфиденциальности данных, сервис целостности данных, сервис единого входа.

Сервис аутентификации (authentication) обеспечивает проверку аутентичности контрагента в процессе взаимодействия и его идентификацию в системе (сопоставление контрагента представляющему его внутрисистемному объекту). Для успешной аутентификации контрагента последний должен обладать некоторой секретной информацией, и должен быть реализован механизм, позволяющий безопасным образом подтвердить наличие у контрагента такой информации. В основном сервис

аутентификации реализуют на основе варианта одного из двух механизмов: разделяемый секрет и асимметричная криптография.

Механизмы безопасности на основе *разделяемого секрета* предполагают наличие у обеих взаимодействующих сторон общей секретной информации. В простейшем случае эта информация хранится и передается открытым текстом. В более сложных реализациях механизма проверяющая сторона не имеет доступа к разделяемому секрету в открытом виде, а хранит значение некоторой функции от него, и сама секретная информация никогда не передается по каналу связи в открытом виде. Примерами реализации механизма разделяемого секрета служат парольная и биометрическая аутентификация.

Механизмы безопасности на основе *асимметричной криптографии* предполагают наличие у аутентифицируемой стороны пары криптографических ключей. Эти ключи связаны определенными математическими отношениями, которые делают статистически невозможным вычисление одного из них при наличии другого. Один из ключей является секретным и известен только его владельцу. Другой ключ пары является открытым и может свободно распространяться. В механизмах безопасности реализуются различные варианты процесса, позволяющего проверить, что контрагент является владельцем парного секретного ключа для заданного открытого. Часто в дополнение к этому механизму используется *инфраструктура открытых ключей* (public key infrastructure – PKI), которая обеспечивает криптографически защищенное сопоставление открытых ключей субъектам взаимодействий.

Сервисы аутентификации могут быть специализированы для поддержки определенных видов взаимодействий и, соответственно, использовать различные механизмы безопасности. Явно выраженной спецификой обладают сервис аутентификации удаленной системы и сервис аутентификации источника данных. Первый предназначен для проверки подлинности системы при взаимодействии с ней в режиме реального времени. Сервис аутентификации источника данных служит для идентификации системы, с которой связаны некоторые данные, например, входящее сообщение.

Сервис контроля доступа (authorization) – это сервис безопасности, предотвращающий использование ресурсов системы вопреки политике безопасности. Существуют две основные разновидности этого сервиса: дискреционный контроль доступа (discretional access control – DAC) и принудительный контроль доступа (mandatory access control – MAC). Дискреционный контроль доступа использует механизм списков

контроля доступа (access control list – ACL). Сервис принудительного контроля доступа реализуется через механизм меток безопасности (label) и уровней доступа (clearance). Сервис контроля доступа должен содержать *монитор обращений* (reference monitor), который представляет собой абстрактную машину, обрабатывающую каждую операцию доступа субъектов системы к ее объектам. Монитор обращений должен быть:

- полным (обрабатывающим каждую операцию доступа)
- изолированным (недоступным для изменений другими сущностями системы)
- проверяемым (достаточно простым и компактным, чтобы имелась возможность подтвердить корректность его работы в соответствии с заданной политикой безопасности)

Сервис аудита (audit) – служба, сохраняющая информацию, необходимую для установления ответственности за события в системе и за вызвавшие их действия системных сущностей. Эта информация должна быть достаточно полной, чтобы по ней можно было проследить действия и установить субъекта, выполнившего их. Данные аудита также должны быть защищены от модификации неавторизованными компонентами системы. Дополнительно записи в журнале аудита могут быть заверены (например, с помощью электронной цифровой подписи) независимой системой, что позволяет повысить юридическую значимость журнала.

Сервис конфиденциальности (confidentiality) и сервис целостности (integrity) данных обычно используются совместно, поскольку каждый сервис, применяемый в отдельности, подвержен определенному классу атак. Сервис конфиденциальности обеспечивает защиту данных от неправомочного раскрытия, обычно путем преобразования данных по криптографическим алгоритмам, но не обеспечивает обнаружение случаев модификации данных. Сервис целостности позволяет обнаружить факты случайных или неправомочных умышленных изменений защищаемых данных, но не подвергает преобразованию сами данные. Сервисы могут использоваться для защиты данных как при их хранении, так и при передачи по каналам связи. Однако, если во время передаче информации по каналу связи при обнаружении неавторизованных модификаций потока данных последние обычно передаются повторно, то для защиты данных от модификаций при хранении сервис целостности играет вспомогательную роль, а основные механизмы защиты должны обеспечиваться, например, непосредственно хранилищем.

Сервис единого входа (single sign-on – SSO) – сервис безопасности, позволяющий пользователю получать доступ к различным системам, аутентифицировавшись один раз.

Для реализации единого входа системы должны установить доверительные отношения так, чтобы они могли оперировать идентификаторами пользователей, обращающихся к системам.

К примеру, государственная организация имеет центральный портал. Ссылки с портала ведут к нескольким подчиненным системам. Пользовательская информация хранится централизовано на основном портале, а подчиненные системы используют широкий спектр систем управления доступом. Технология единого входа позволяет пользователям аутентифицироваться один раз на центральном портале и затем обращаться к подчиненным системам без необходимости повторно вводить имя/пароль.

Рисунок 6. Схема работы сервиса единого входа

2.5.3Выводы

Разработка безопасности системы должна начинаться с определения, что нужно защищать, от каких угроз необходима защита, и выработки соответствующей политики безопасности, которая должна включать как технические, так и организационные мероприятия. Далее, на основании политики, выбирается модель безопасности, определяются необходимые сервисы безопасности, разрабатывается архитектура и реализуются необходимые механизмы и сервисы. На всех уровнях процесса нужно учитывать требования и возможности интеграции разнородных систем.

2.6 Транзакции

2.6.1 Постановка задачи

Большинство систем управления автоматизированными рабочими процессами нуждаются в поддержке транзакционности для того, чтобы в качестве результата своей работы выдавать взаимно-согласованные всеми участниками процесса данные. Поддержка транзакционности гарантирует, что получаемые результаты сохраняют свою целостность на пути исполнения всех заданий в пределах всего бизнес-процесса. При этом результаты выполнения отдельного задания обычно становятся доступными до завершения всего процесса. Например, система бронирования билетов на самолет может зарезервировать место на определенный период времени, но если пассажир не подтвердит свой заказ в течение отведенного временного интервала, заказ будет переоформлен на другого пассажира.

Транзакции своеобразным «фундаментом» для построения надежных распределенных приложений. Транзакции являются механизмом, который позволяет нам быть уверенными в том, что все участники распределенного взаимодействия пришли к взаимно-согласованному результату. Традиционно, транзакции характеризуются следующими свойствами, которые в совокупности называются ACID-моделью транзакций:

- Атомарность: в случае успешного выполнения транзакции выполняется каждая из входящих в нее операций, а в случае неуспеха – ни одна из них.
- Связность: приложение выполняет корректное изменение системы по завершению.
- Изоляция: результат выполнения операций внутри транзакции становится доступен внешним системам только после ее успешного завершения.
- Долговечность: если транзакция завершилась успешно, её результат будет зафиксирован и сохранён.

Трудно, если вообще возможно, применить традиционную архитектуру транзакций в среде взаимодействия слабосвязанных распределенных приложений, взаимодействующих в рамках бизнес-процесса. Более того, большинство систем управления автоматизированными потоками работ поддерживают сложные, долгоживущие процессы, где может понадобиться компенсация выполненных действий, а не полный их откат, или выбор другого приемлемого пути выполнения процесса.

Традиционные системы управления транзакциями обеспечивают поддержку транзакций, если задача приложения может быть целиком представлена как одна транзакция верхнего уровня. Однако, ото очень часто не так. Транзакции верхнего уровня

наиболее удобно рассматривать как сущности с коротким временем жизни, которые осуществляют стабильные изменения состояния системы. Они гораздо меньше подходят для построения долгоживущих приложений, которые выполняются минуты, часы, дни или даже в течение большего периода времени. Долгоживущие транзакции верхнего уровня могут снизить могут снизить возможность параллельного исполнения задач в системе до недопустимого уровня, блокируя ресурсы на длительные промежутки времени. Более того, если транзакция отменяется, многие уже выполненные ценные действия будут потеряны.

2.6.2Обзор и анализ

Среди основных моделей реализации механизма транзакций в среде распределенных приложений выделяется модель атомарных транзакций и модель бизнес-транзакций.

Модель атомарных транзакций определяет специфический набор протоколов, которые используются для реализации протоколов традиционной двухфазной атомарной транзакции. Важно отметить, что атомарная двухфазная модель касается только вовлеченных сервисов. Сторона, предлагающая сервисы, может объявить успешное двухфазное завершение, но использовать некоторую другую, внутреннюю для предприятия, модель, типа компенсации или версионности. Эта свобода делает простую модель двухфазного успешного завершения полезной для продолжительных Интернет вычислений.

Модель бизнес-транзакций определяет специфический набор протоколов, которые используются для реализации продолжительных, основанных на компенсациях протоколов транзакций.

Когда определен бизнес процесс и его связи с бизнес-партнерами, следующим шагом необходимо предоставить механизм, координирующий все задачи, включенные в процесс, для того чтобы мог быть получен единый надёжный результат. Обычно, есть несколько различных задач, которые должны быть удачно выполнены, некоторые задачи должны выполняться одновременно, и весь процесс должен иметь сведения о том, выполнились все задачи удачно, или произошел сбой.

Распределенные, выполняющиеся длительное время бизнес-транзакции всегда были трудно решаемой проблемой. Помимо базовых технических вопросов, большая продолжительность и задержки связи, вносимые набором бизнес-задач, выполняемых различными бизнес-партнерами, использующими потенциально несовместимые программные системы, вносит сложные требования. Транзакции должны управляться по

принципу «задача-за-задачей», с мониторингом инфраструктуры и управлением результатами каждой задачи по отдельности. Традиционные транзакционные и координирующие системы наилучшим образом подходят для управления индивидуальной бизнес-задачей.

2.6.3Выводы

При реализации продолжительных по времени рабочих процессов (а именной такой спецификой и обладает большинство систем поддержки управления взаимодействием распределенных информационных систем) модель бизнес-транзакций является наиболее подходящим механизмом обеспечения согласованности и целостности результата выполнения сложного бизнес-процесса. К основным достоинствам данного подхода относятся:

- возможность участвующим в процессе ИС сразу иметь доступ к результатам выполнения некоторой операции транзакции, что необходимо для согласованного взаимодействия;
- отсутствие необходимости в длительных блокировках общих ресурсов процесса, так как в бизнес-процессах такие блокировки могли бы продолжаться недопустимо длинный период времени;
- ослабление свойства атомарности: механизм компенсации бизнес-транзакций позволяет в случае возникновения исключительной ситуации не отменять все проделанные до данного момента действия, а произвести логическую компенсацию. При этом процесс может быть просто перенаправлен по другой ветви, а результаты всех важных действий, совершенных к текущему моменту, будут сохранены.

Ввиду этого в качестве модели транзакций в рамках СУВ рекомендуется использовать модель долгоживущих бизнес-транзакций.

2.7 Рабочие процессы

2.7.1Постановка задачи

2.7.1.1Перспективы использования управляемых рабочих процессов для автоматизации электронных взаимодействий

За последние 15 лет были разработаны средства, позволяющие не только выполнять конкретные работы, но и управлять их потоками (Workflow). Эти процессы

формально специфицируются в компьютерных системах. Потоки работ управляются компьютерной программой, которая назначает задания, принимает их и фиксирует степень их исполнения. Традиционно понятие «рабочий процесс» определяется с позиций офисных работ, работ с документами – передача документа, подписание приказа, подготовка накладной. Но те же самые принципы и понятия встречаются, например, в работе промышленного предприятия – подготовка документов, деталей, станков и рабочего персонала для сборки сложных технологических систем.

С появлением сложных Web-систем, цифровых библиотек, корпоративных порталов, поддерживающих и управляющих большими объемами и потоками информации, сформировалось категория подсистем, называемых системами управления содержанием Web-систем (WCMS - Web Content Management System). Первые WCMS представляли совокупности Web-форм для управления данными – как несвязанных Web-форм, так взаимосвязанных Web-форм, но с фиксированным в программном коде порядком обработки. Сейчас актуальной становится потребность в более гибкой организации управления потоками работ, в поддержке декларативных форм определения и оперативного изменения потоков работ. WCMS должны управлять созданием и модификацией информационных, сильно взаимосвязанных ресурсов произвольных типов. Потоки работ могут быть связаны не только с поддержкой информационных ресурсов, но и с выполнением полностью автономных регламентных процедур системы в ответ на, например, программные обращения, на наступление некоторого события и т.п.

Преимущества автоматизированных систем управления потоками работ:

- Работа не застаивается и выполняется наиболее подготовленными сотрудниками – «контролерам» процессов все реже приходится исправлять ошибки, возникающие в ходе работ.
- Все процедуры формально специфицированы и точно выполняются, что гарантирует выполнение работы в соответствии с планом и всеми деловыми требованиями.
- На каждое задание назначается наиболее квалифицированный человек (или более «интеллектуальный» автомат), и самые важные задания выполняются в первую очередь. Пользователи больше не задумываются о том, чем им заниматься, возможно, откладывая важную, но сложную работу.
- Параллельная обработка, когда одновременно выполняются два или более заданий, является более широко используемым подходом, чем традиционный, ручной процесс.

Когда наиболее квалифицированное лицо выполняет наиболее важную работу, следуя правильным процедурам, происходит не только улучшение выполнения процесса в целом, но и снижение затрат на производство, равно как повышение уровня услуг, предоставляемых потребителю. Уверенность в том, что работа распределяется справедливо и включает в себя только «правильные» действия, улучшает настроение работника. Таким образом, регулируемые потоки работ приносят пользу с позиций всех трех участников бизнес-процесса – компании, рабочего персонала и потребителя.

2.7.1.2Задача стандартизации языка описания автоматизированных рабочих процессов (АРП)

На данный момент на рынке программных продуктов сложилась непростая ситуация со стандартизацией процесса построения систем управления потоками работ, и она имеет пока скорее теоретический, чем практический характер.

Между тем, следование каноническим спецификациям и стандартам дает ряд очевидных преимуществ как в случае систем управления потоками работ, так и в случае других практических решений:

- Строгая формализация процесса описания потоков работ: можно реализовать стандартный (возможно частично автоматизированный) процесс по анализу и построению новых моделей реальных рабочих процессов.
- Переносимость и интероперабельность: модели процессов, созданные в рамках одной системы, могут частично или полностью работать под управлением другой (возможно, удаленной) системы.
- Универсальность: расширяемость стандартов делает возможным применение единого механизма описания управления потоками работ в различных сферах деятельности.

Таким образом, следование ряду стандартов позволяет создавать автономные, открытые системы управления потоками работ, что, безусловно, расширяет область и способы их применения для моделирования различных рабочих процессов.

2.7.2Обзор и анализ

Среди основных подходов к описанию рабочих процессов выделяются графовая и блочная модель описания рабочих процессов.

К характерным особенностям графового подхода можно отнести следующее:

- рабочий процесс представляет собой направленный граф, узлами которого являются «действия», связанные между собой переходами. Переходы могут быть

условными, причем условие проверяется на этапе выполнения конкретного «действия»;

- между последовательно выполняющимися «действиями» может быть организован поток данных для обмена управляющей и другой информацией;
- допускается возможность параллельного выполнения нескольких «действий»;
- существует возможность выделения «блоков» – возможность объединения «действий» в блок «действий» со своими отдельными условными или безусловными точками входа и выхода;
- имеется возможность определять вложенные подпроцессы внутри родительского процесса, которые сами по себе представляют полноценные потоки работ.
- элементы описания процессов имеют обширный набор атрибутов, определяющих ход выполнения процесса. К ним можно отнести условные выражения для переходов, временные рамки, задание множественных исполнителей «действий» и т.д.;
- существует два выделенных типа «действий»: начальные (на которые не осуществляются переходы) и конечные (которые не связаны «исходящими» переходами с другими «действиями» процесса).

К характерным особенностям блочного подхода можно отнести следующее:

- элементарными блоками, из которых строится модель рабочего процесса, являются *«действия»*;
- с позиций блочного подхода сам рабочий процесс не выделяется в виде отдельного объекта, а является просто специальным случаем агрегированного действия;
- существует понятие вложенного процесса, образующего свой собственный контекст;
- существует понятие процесса, обрабатывающего исключительные события;
- существует понятие процесса-компенсатора, который позволяет проводить действия по освобождению системных ресурсов и другие завершающие шаги.
- между последовательно выполняющимися «действиями» может быть организован поток данных для обмена управляющей и другой информацией посредством определения понятия переменных процесса с их областями видимости в пределах соответствующих контекстов (вложенных процессов);
- допускается возможность параллельного выполнения нескольких «действий» с определением порядка их запуска;

- существует возможность определения агрегированных блоков «блоков» для композиции «действий» процесса;
- имеется возможность определять вложенные подпроцессы внутри родительского процесса, которые сами по себе представляют полноценные потоки работ;
- элементы описания процессов имеют обширный набор атрибутов, определяющих ход выполнения процесса;
- структура процесса аналогична структуре программ, написанных на языках программировании высокого уровня.

2.7.3Выводы

Графовый подход к описанию потоков работ более нагляден и всегда может быть изображен в виде направленного графа переходов между «действиями». При этом с точки зрения выразительности он лишен многих достоинств блочного подхода, обусловленного модульной архитектурой последнего. К основным достоинствам блочного подхода перед графовым, определяющим его выбор для формализации языка описания рабочих процессов можно отнести:

- возможность определения обработчиков исключительных ситуаций;
- возможность использования механизмов компенсации
- возможность определения эффективного потока данных между действиями процесса посредством стандартного механизма переменных с их областями видимости
- механизм определения вложенных контекстов с их областями действия аналогичный механизму подпрограмм языков высокого уровня.

2.8 Глобальная идентификация

2.8.1Постановка задачи

При организации распределенной системы необходимы механизмы именования ресурсов, обеспечивающие глобальную уникальность имен ресурсов, их независимость от местоположения ресурсов. Это требуется при организации взаимосвязей ресурсов, агрегировании результатов распределенных запросов, используется при индексировании, репликации и т.п. Уникальный идентификатор ресурса автоматически сопоставляется ресурсу при его создании или регистрации в распределенной среде. Уникальные имена ресурсов, в частности, обеспечивающие извлечение ресурсов, имеются в любой локальной системе. В распределенной системе необходим общий механизм

идентификации. В обязанности локальной системы входит связывание глобального имени с локальным.

В основе задачи уникальной идентификации ресурсов лежат проблема гарантирования бессрочного существования имени. Какая форма именования должна быть создана, чтобы она могла использоваться всегда? С технической точки зрения понятие “всегда” не существует. Поэтому можно говорить о фиксированном сроке службы имени и механизме, который будет обеспечивать переход ресурсов и имен в новую форму прежде, чем этот срок службы истечет. Механизм должен быть в состоянии обеспечивать модификацию идентифицируемых ресурсов по мере перехода к новой форме, что на данный момент не представляется возможным.

Сейчас можно говорить только о поддержке служб, которые возьмут на себя задачи обеспечения выдачи и разрешения уникальных имен, которые займутся разработкой механизмов для облегчения перехода к новым формам вместе с развитием технологий.

В 1994 г были сформулированы IETF принципы универсального имени ресурса (URN - uniform resource name) призванного идентифицировать информационные ресурсы вне зависимости от их местоположения в Интернет. URN является постоянным и уникальным идентификатором. URN может быть присвоен только ресурсу, являющемуся достаточно “стабильным” и имеющему “существенное” интеллектуальное, информационное содержание. Будучи сопоставленным ресурсу, он не может быть изменен. Копии ресурса должны использовать один и тот же URN вне зависимости от формы представления информации. “Тот же” ресурс может получить новый URN, только если его интеллектуальное содержание изменилось, но, следовательно, это уже “новый” ресурс. “Старый” ресурс может быть сохранен и быть доступным по старому URN. Он может быть удален, но его “использованный” URN не может быть использован для другого ресурса.

Использование общепризнанного стандарта позволяет избавиться от частных схем именования и сопутствующих им проблем переименования, обеспечить интеграцию информации в Интернет и поддержу ее актуальности. Чтобы поддержать существующие общепризнанные и специализированные схемы именования источников информации, URN вводит понятие пространства имен (name space) как именованной совокупности имен, соответствующих некоторой зарегистрированной схеме. Для обеспечения понятия уникального имени используются системы регистрации пространства имен, формирования и разрешения имен. Первая обеспечивает международную регистрацию схем именования. Вторая специализируется на конкретной схеме и отвечает за поддержку уникальности имен. Третья должна сообщать по URN сведения об именуемом ресурсе, в которые могут быть включены данные о место положениях ресурса, о способе

доступа к нему по сетевым протоколам. В Интернет среде в качестве источников таких сведений рассматривается URL. На текущий момент имеется ряд реализаций систем именования и их поддержки в Интернет, например, Handle System, DOI, причем вторая реализована на основе первой и сопровождается организационно административной структурой.

Глобально уникальная идентификация ресурсов применяется для решения следующих задач:

- Интеграция ресурсов: различные информационные системы могут содержать частичную информацию об одном и том же ресурсе. Должна существовать возможность установления факта принадлежности информации, полученной от различных систем, одному ресурсу.

Методика глобально уникальной идентификации интегрируемых ресурсов должна обеспечивать выполнение следующих требований:

- Глобальность: возможность ссылаться на нужный ресурс из любого места среды
- Уникальность: одно имя не может быть присвоено различным ресурсам
- Постоянство: срок жизни идентификатора не ограничен сроком жизни ресурса
- Имена могут присваиваться любым ресурсам, независимо от их типа
- Возможность получения информации или метаинформации о ресурсе в том случае в тех случаях, когда это имеет смысл
- Идентификатор может быть создан приложением из любого места среды
- Безопасность функционирования (возможность использования защищённых протоколов, аутентификация и авторизация клиентов системы)

2.8.2Обзор и анализ

Ниже приведена суммарная таблица по существующим системам идентификации.

Таблица 2. Суммарные сведения по существующим системам идентификации

Категория	**Спецификация**
Постоянные и уникальные логические идентификаторы	Стандарт **ANSI/NISO Z39.84** определяет синтаксис для глобальной идентификации электронных данных Примечание: необходим механизм для поддержки уникальности

Постоянные идентификаторы	**DOI** предоставляет способ присваивания идентификаторов электронным данным
	XRI (OASIS Extensible Resource Identifier) Целью XRI является специфицировать URI схему и соответствующее пространство имён URN для распределённых служб каталогов, которая позволит идентифицировать ресурсы (в том числе людей и организации) и разделять данные между компаниями и приложениями.
Уникальные идентификаторы	**GUID (Globally Unique Identifier)**, не существует правил синтаксиса для GUID. Приложения должны обрабатывать его как строку текста. Является частью стандарта RSS 2.0.
Постоянные идентификаторы	Возможно использование National Bibliography Number (NBN) как **Uniform Resource Names (RFC 3188)**
Система разрешения идентификаторов	**Handle system** является системой разрешения и системой именования. Идентификаторы создаются в соответствующих областях ответственности, и не имеют предписанного синтаксиса.
Идентификаторы для постоянных URL	**PURLs (persistent URL)** функционально являются URL. Однако, они указывают не на фактическое местоположение ресурса, а на промежуточные сервис разрешения.
Постоянные имена для URL	**URN (Uniform Resource Number).** URN представляют собой постоянное, глобально-уникальное имя, назначенное объекту. В противоположность URL, который меняется каждый раз при смене месторасположения объекта, URN не зависит от месторасположения и, как следствие, у него большее время жизни.

Зарегистрированные пространства имён	**URI (Uniform Resource Identifier)** являются зарегистрированными идентификаторами, указывающими на протоколы или пространства имён. URN представляют собой форму URI, которая использует пространство имён для постоянных имён объектов.
Схемы для идентификации ресурса в Интернет	**URL (Uniform Resource Locator)** – это адрес ресурса, доступ к которому можно получить через Интернет. URL должен содержать достаточную информацию для нахождения объекта по указанной схеме. В случае URL, указывающих на ресурс, доступный по HTTP, схемой является “http”, а схемозависимая часть содержит адрес сервера и путь документа на этом сервере.
Идентификаторы для электронных объектов посредством ASN.1	**OID** используются в протоколах, основанных на ASN.1
Архивные идентификаторы	**ARK (Archival Resource Key)** представляет собой схему, которая предназначена для упрощения присваивания постоянных идентификаторов и получения информационных объектов.

2.8.2.1Абстрактные форматы: URI (URL, URN, URC), XRI

Uniform Resource Identifier (URI) разрабатывался как стандарт, который утверждает концепции и синтаксис для различных стандартов, применимых для наименования, описания и указания на сетевые ресурсы. В их число входят: URL (Uniform Resource Locator) – стандарт для указания на положение сетевого ресурса, URN (Uniform Resource Name) – стандарт для уникального именования цифровых ресурсов, и URC (Uniform Resource Characteristics) – метаданные о ресурсе.

Функциональными требованиями для URN являются:

- Глобальная область действия: имя имеет одно и то же значение в любом месте
- Глобальная уникальность: одно и то же имя никогда не будет присвоено различным ресурсам
- Постоянство: срок существования имени неограничен, независимо от срока существования ресурса

- Масштабируемость: имена могут быть присвоены любым ресурсам
- Возможность расширения в будущем
- Независимость: организация, выдающая имена, самостоятельно устанавливает правила такой выдачи

Идентификаторы URN выглядят следующим образом:

URN:<NID>:<NSS>

Здесь NID (Namespace Identifier) – идентификатор пространства имён, а NSS (Namespace Specific String) – строка, интерпретация которой зависит от пространства имён. Каждое пространство имён определяет структуру. Пример идентификатора:

URN:ISBN:0872783665

Такие идентификаторы могут храниться на сетевом сервисе для “разрешения” в фактические адреса ресурсов. В процессе разрешения пользователь посылает запрос на сервер, и в результате запроса сервер посылает пользователю список URL адресов, соответствующих ресурсу.

Система доменных имён Интернета (DNS) работает похожим образом, однако, в отличие от DNS, URN позволяет поддерживать несколько схем именования. Для того чтобы поддерживать произвольное количество схем именования, поддержка пространств имён и сервис разрешения должны быть независимы. Поэтому, требуется наличие дополнительного сервиса – реестра URN. Данный сервис должен содержать информацию о схемах именования, системах разрешения, и может указать пользователю, какой сервер разрешения может обработать заданный URN.

Стандарт XRI (Extensible Resource Identifier), разработанный OASIS, похож на URI, но содержит дополнительные синтаксические элементы и позволяет использовать в идентификаторах национальные символы.

XRI расширяет синтаксис URI следующим образом:

- Постоянные и переназначаемые фрагменты. Общий синтаксис URI не различает постоянные и переназначаемые идентификаторы. Синтаксис XRI позволяет любой фрагмент идентификатора выразить либо как постоянный, либо как переназначаемый.
- Неограниченное делегирование – синтаксис XRI поддерживает делегирование как постоянных, так и переназначаемых идентификаторов в любом фрагменте пути.
- Перекрёстные ссылки. XRI идентификаторы могут включать другие идентификаторы (URI или XRI) как часть.
- Интернациональный набор допустимых в идентификаторах символов.

- Символы глобального контекста.
- Любой идентификатор может быть выражен как неразрешаемый. Такие идентификаторы характеризуют не какой-либо ресурс, а абстракцию, и могут применяться, например, для установления эквивалентности.

2.8.2.2The Handle System

The Handle System является разработкой CNRI (Corporation for National Research Initiatives) и в целом соответствует требованиям к URN. Разработка определяется как “всесторонняя система для присваивания, управления и разрешения постоянных идентификаторов для цифровых объектов и других ресурсов в Интернет”. Идентификаторы этой системы разрешаются с помощью глобального сервиса. Существует возможность использования локального сервиса разрешения, интегрированного в глобальную систему.

“Handle System” состоит из открытого протокола, пространства имён, и реализации протокола. Протокол позволяет распределённым компьютерным системам сохранять имена цифровых ресурсов и разрешать эти имена в информацию достаточную для нахождения, доступа и других использований ресурса. Эти значения, ассоциированные с идентификатором, могут быть изменены по необходимости, для того чтобы соответствовать текущему состоянию ресурса без смены идентификатора. Это позволяет идентификатору оставаться постоянным в процессе изменения расположения ресурса и других его характеристик. У каждого идентификатора могут быть собственные администраторы, и администрирование может производится в распределённой среде. Система поддерживает безопасное разрешение идентификаторов. Сервисы безопасности, такие как конфиденциальность и целостность данных, предоставляются по клиентскому запросу.

The Handle System разработана с учётом следующих целей:

- Уникальность: каждый идентификатор является глобально уникальным в рамках системы
- Постоянство: идентификаторы могут быть использованы как постоянные для Интернет-ресурсов. Имя идентификатора не обязано основываться на ресурсе, который он идентифицирует. Даже несмотря на то, что в качестве имени можно использовать текущее имя ресурса (например, для удобства), единственная связь между идентификатором и идентифицируемым ресурсом содержится внутри системы. Естественно, ничто не гарантирует постоянную доступность ресурса по данному имени – такое постоянство обеспечивается административно. Система

лишь предоставляет возможность одному и тому же идентификатору оставаться актуальным при смене расположения ресурса, владельца ресурса, или других его свойств.

- Один и тот же идентификатор может ссылаться на несколько экземпляров ресурса, распределённых в сети. Это свойство может быть использовано приложениями для повышения надёжности и производительности.
- Идентификатор может описывать несколько атрибутов ресурса, в том числе связанные с ресурсом сервисы, доступные с помощью любых методов по различным и, возможно, изменяющимся, адресам.
- Поддержка интернационализации: пространство имён основано на Unicode 3.0. Это позволяет присваивать идентификаторам имена, содержащие национальные символы. Протокол использует кодировку UTF-8 для работы с идентификаторами.
- Распределённая модель сервисов: система определяет иерархическую модель сервисов. Каждое локальное пространство имён может обслуживаться соответствующим локальным сервисом, глобальным сервисом, или и тем и тем. Глобальный сервис (Global Handle Registry) используется для перенаправления запросов на разрешение идентификаторов на соответствующие локальные сервисы. Распределённая модель позволяет реплицировать идентификаторы любого пространства имён на несколько серверов. Каждый сервис может быть распределён по нескольким физическим компьютерам.
- Безопасность: протокол предоставляет возможность безопасного разрешения идентификаторов и администрирования поверх сети Интернет. Протокол определяет стандартные механизмы аутентификации клиента и сервера, а так же авторизацию сервисов. Предоставляются возможности по поддержке целостности данных и конфиденциальности.
- Распределённое администрирование: у каждого идентификатора может быть собственный администратор или группа администраторов. В конечном итоге это потенциально позволяет управлять любым идентификатором из любого места сети.
- Эффективный сервис разрешения: протокол разработан с учётом возможности высокоэффективного разрешения. В частности, сервер может разрешать идентификаторы и обрабатывать административные запросы с использованием разных процессов и портов, благодаря чему на разрешение не будут влиять более медленные административные операции.

2.8.2.3DOI

DOI (Digital Object Identifier) – система для постоянной идентификации и обмена интеллектуальной собственностью в цифровых сетях. Предоставляется расширяемая среда для управления интеллектуальной собственностью в любой форме, на любом уровне и в любом цифровом окружении. Разработкой, политиками и лицензированием данной технологии управляет International DOI Foundation.

2.8.2.4UUID

UUID представляет собой всемирный “уникальный идентификатор”. Это 128-битные числа, присваиваемые объектам, которые гарантированно являются уникальными. Алгоритм, который применяется для гарантии уникальности, комбинирует в идентификаторе адреса оборудования, время генерации и случайные значения.

В UUID содержится адрес первой сетевой карты вычислительной системы, которая сгенерировала данный идентификатор. Данный фрагмент обеспечивает уникальность идентификатора в пространстве, поскольку MAC адреса каждой сетевой карте присваиваются из единого источника, при этом гарантируется их уникальность.

2.8.2.5PURL

PURL (Persistent Uniform Resource Locator) является сервисом именования и разрешения Интернет ресурсов. Он предназначен как временное решение для использования, пока URN не являются реализованными. PURL выглядит аналогично URL, с тем исключением, что указывает не на ресурс, а на сервис разрешения. Сервис перенаправляет пользователя на реальный URL адрес. В случае смены адреса ресурса, достаточно прописать его новое местоположение в системе разрешения PURL, при этом пользователям он будет доступен под тем же PURL именем.

2.8.3Выводы

Из существующих в настоящий момент реализаций наиболее близко предъявленным требованиям соответствует The Handle System. Остальные технологии либо ориентированы на какую-либо предметную область (DOI, PURL), либо не описывают технологию доступа к метаинформации (UUID, абстрактные форматы). В то же время при внедрении этого решения надо принимать во внимание следующие обстоятельства:

- Для корректного функционирования клиентского программного обеспечения сервер должен быть зарегистрирован в глобальном реестре, что в настоящий момент осуществляется вручную операторами глобального реестра по письменному запросу. При переносе сервера так же следует уведомлять об этом оператора. В

результате, при установке и смене конфигурации могут возникать задержки. Однако, поскольку протоколы Handle System являются открытыми, для обеспечения возможности автономных испытаний может оказаться оправданным создание собственной реализации клиентского программного обеспечения, не взаимодействующего с глобальным реестром.

- Серверное и клиентское программное обеспечение имеет ограниченную лицензию на использование, в результате чего может возникнуть потребность лицензирования технологии

3 Обоснование обеспечения интеграции с применением Web-сервисов (Архитектуры WSA)

3.1 Постановка проблемы

В рамках данного раздела рассматриваются основные варианты интеграции ИС на основе СУВ и выбора наиболее подходящей модели архитектуры для ее применения в качестве стандартной при интеграции ИС.

При этом определяется, что есть критерии к группам механизмов, описанным в предыдущем разделе, и делается выбор той архитектуры, которая в наибольшей степени удовлетворяет сформулированным критериям предыдущего раздела.

3.2 Обзор и анализ архитектур для интеграции ИС

Топология интеграционного решения отражает различные способы соединения взаимодействующих ИС и интеграционных компонентов. Среди основных топологий выделяются соединения типа «точка-точка», шлюзы и шины.

В соединении «точка-точка» интегрируемые ИС устанавливают прямые соединения друг с другом. Простота метода осложняется необходимостью для взаимодействующих ИС знать точную адресную информацию о партнерах, уметь осуществлять преобразования между используемыми партнерами форматами сообщений. Интеграция данного типа требует разработки дополнительного и изменение существующего программного кода взаимодействующих ИС и приложений.

Механизмы адаптеров и коннекторов позволяют осуществить преобразование специфических интерфейсов и данных конкретной ИС в интерфейсы и данных стандартной интеграционной среды. Прикладные программные интерфейсы позволяют встроить в ИС код, позволяющий работать с транспортной инфраструктурой. Настройка интерфейсов, адресной информации и процедур трансформации, применяемых в адаптерах вместо написания кода, позволяет ускорить внедрение интеграционного решения.

Следующий важный компонент интеграционной среды - интеграционный сервер (брокер), шлюз или система управления взаимодействиями (СУВ), выступающий как центральный узел (или для обеспечения требуемой масштабируемости совокупность взаимосвязанных узлов), к которому направлены обращения интегрируемых ИС и

приложений. Через СУВ направляются потоки сообщений и данных с этих ИС; СУВ перераспределяет, обрабатывает и направляет эти потоки. Использование интеграционного сервера позволяет возложить на него функции адресации и трансформации передаваемых данных, причем гораздо более сложные, чем в случае прямого соединения или использования адаптеров. Централизованное исполнение функций обработки передаваемой информации и данных означает использование общего репозитория для корпоративных интеграционных правил. Правила из общего репозитория прозрачны для разработчиков и системных администраторов, могут дополняться и изменяться при изменении деловых процессов и состава вычислительной среды, что позволяет эффективно соединять и динамически управлять интеграционным решением.

Благодаря простоте, надежности и универсальности программное обеспечение систем очередей сообщений является одной из перспективных технологий интеграции. Системы очередей сообщений предоставляют асинхронный метод взаимодействия для прикладных программ. Приложения обмениваются данными, посылая и принимая их в виде сообщений. При этом приложения не взаимодействуют друг с другом напрямую, а обращаются к службе очередей сообщений и при помощи API получают доступ к очередям и сообщениям. Очереди представляют собой промежуточное место хранения сообщений, своего рода специализированную базу данных с механизмами, гарантирующими сохранность данных: журналирование, восстановление после сбоев, обработка транзакций.

В распределенной среде за передачу сообщений между системами отвечают сетевые компоненты системы очередей сообщений. При использовании протокола гарантированной передачи посылаемое сообщение не будет удалено из очереди на сервере посылки до тех пор, пока оно не будет полностью принято на сервере-адресате — при передаче сообщений реализуется сетевая транзакция.

В течение целого ряда лет развиваются и совершенствуются различные компонентные прикладные архитектуры CORBA, J2EE, .Net. Можно указать на два момента, касающихся отношений подобных архитектур и решений на базе систем передачи сообщений. С одной стороны, они не являются взаимоисключающими: системы передачи сообщений могут эффективно сочетаться с компонентными архитектурами, привнося функции гарантированной доставки и асинхронного взаимодействия для взаимодействия распределенных компонентов. С другой стороны, в реальной жизни постоянно соседствуют многочисленные системы, построенные на различных архитектурах, не адаптируемые под спецификации и стандарты конкретной компонентой

архитектуры. Практика показывает: для взаимодействия между подобными слабосвязанными системами наиболее пригодны простые интеграционные решения на принципах обмена сообщениями.

Среди преимуществ системы очередей сообщений можно отметить многообразие сценариев взаимодействия программ — от простой рассылки информации адресатам до поддержки контентной адресации в модели «публикация-подписка» и сложных распределенных и распараллеленных вычислительных процессов. Кроме того, существование дополнительного звена в виде очередей сообщений позволяет вставлять промежуточную обработку передаваемой информации для решения различных задач: обеспечение безопасности и совместимости форматов, динамической маршрутизации, анализа контента и т.д.

Интеграционные сервера обеспечивают взаимодействие приложений через центральный промежуточный шлюз, поддерживающий интеллектуальную обработку и распределение данных между приложениями. Роль брокера в интеграционной среде, основанной на передаче сообщений, — это роль почтамта, сортирующего и регистрирующего почтовые отправления, проверяющего адресную информацию и распределяющего их для дальнейшей отправки адресатам.

3.3 Выбор архитектуры WSA

Архитектура Web Services Architecture (WSA) обеспечивает полную независимость от платформы реализации, облегчая тем самым процесс межсистемной интеграции и уменьшая трудности взаимодействия для пользователей. Web-сервисы потенциально могут обеспечить сокращение сроков и затрат на внедрение новых приложений, интеграцию существующих.

В частности, целью технологии Web-сервисов является ликвидация многих ограничений, накладываемых современными IT-технологиями на взаимодействие организаций, и, как следствие этого, повышение качества бизнес-процессов, что в итоге позволяет достичь лучших конечных результатов.

WSA дает широкие возможности при определении того, какие из своих служб можно сделать доступными для клиентов, партнеров и поставщиков, клиенты, партнеры и поставщики получает в свое распоряжение более набор предлагаемых сервисов. При создании WSA-систем отдельные сервисы могут строиться с использованием одной или более из этих технологий. Сервис-ориентированная архитектура отличается от ОО и

процедурных систем в одном ключевом аспекте: связывании. Взаимодействие сервисов основывается на том, какие они предоставляют функции, и как они их предоставляют.

3.3.1Экономические преимущества выбора архитектуры WSA

Использование Web-сервисов повышает эффективность деятельности организаций за счет сокращения сроков снижения затрат на внедрение новых приложений, обеспечения широкого внедрения Web-технологий и потенциального роста доходов. В большинстве случаев ключевым фактором является время выхода на рынок. Каждую компанию привлекает возможность быстро и без особых затруднений предоставлять онлайновые сервисы сотрудникам, клиентам, партнерам и поставщикам, используя простые интерфейсы для взаимодействия между разнородными системами. Кроме того, высоко оценивается предоставление внешним контрагентам возможности управления своей финансовой информацией на принципах самообслуживания. Компании-клиенты стремятся с помощью Web-сервисов получить преимущества перед конкурентами и сегодня находятся на стадии получения положительных результатов.

Уже на первых этапах использования WSA многие компании отмечают снижение затрат на центры обработки телефонных вызовов, поддержку клиентов и создание приложений, а также сокращение сроков доставки продукции и/или услуг. Эти компании смогли сформулировать несколько важных преимуществ новой технологии, включая нижеперечисленные:

- Сокращение времени на развертывание новых приложений
- Повышение продуктивности за счет расширенного использования Web-технологий
- Повышение эффективности и снижение затрат
- Повышение доходов

Архитектура WSA оказывает непосредственное влияние на следующие области:

- **Автоматизация бизнес-процессов.** Назначение Web-сервисов - улучшение электронного взаимодействия. Web-сервисы исключают необходимость выполнения отдельного интеграционного проекта для каждого партнера, клиента или поставщика, с которым компания собирается проводить автоматизированные бизнес-операции.
- **Интеграция корпоративных приложений (EAI).** Современные EAI-системы обычно представляют собой специализированные решения, предназначенные для интеграции конкретных приложений друг с другом. WSA является общим решением данной задачи.

- **Интеграция по схеме "бизнес-бизнес" (B2B).** Web-сервисы упрощают взаимодействие с другими компаниями и объединяют собственные функции компании или функции, обеспечиваемые другими участниками бизнес-процесса. Компании получают дополнительную возможность осуществить аутсорсинг (использование внешних сервисов) не только IT-функций, но и целых направлений деятельности сторонним специализированным компаниям. Это позволяет компании сократить штаты и сконцентрироваться на профильных видах деятельности.
- **Конструирование приложений и гибкий подход к аутсорсингу**. Компании получат большую свободу выбора при конструировании приложений. Вместо выбора между обслуживанием всей IT-системы собственными силами и обращением к внешнему поставщику услуг доступа к приложениям, который будет обеспечивать все или большую часть необходимых IT-сервисов, компания получает возможность создания смешанной структуры из собственных и внешних сервисов при неограниченной свободе их комбинирования.
- **Улучшение доступа к бизнес-функциям.** Компания может предоставить внешним пользователям доступ к функциям, первоначально разработанным для внутреннего использования. Доступ может осуществляться как на коммерческой основе, так и бесплатно – в интересах укрепления деловых взаимоотношений.
- **Свобода выбора наилучшей технологической платформы для каждой конкретной ситуации.** Выбор наиболее подходящей платформы для развертывания новой бизнес-функции часто ограничивается необходимостью интеграции с существующими системами, как собственными, так и, во все большей степени, системами бизнес-партнеров. Web-сервисы устраняют указанные ограничения, обеспечивая полную независимость от платформы реализации.
- **Независимость от местоположения и используемых устройств доступа.** Web-сервисы не зависят от типа устройства доступа, технологии подключения или местоположения пользователя. В них легко могут быть интегрированы и/или использованы мобильные подключения, портативные устройства, широкополосный или узкополосный доступ и другие подобные технологии.

Многие компании используют Web-сервисы как средство интеграции с партнерами по электронным операциям. Важный для деятельности организации проект обычно сводится к развертыванию нового приложения, совместно используемого партнерами вне зависимости от различий в применяемых аппаратных и программных платформах. В

настоящее время к приложениям, совместно используемым бизнес-партнерами, а следовательно в первую очередь выигрывающим от применения WSA–технологии, относятся:

- Общее управление доступом
- Интерфейсы прикладного программирования для систем управления знаниями
- Ведение отчетности
- Ведение счетов
- Управление налогами
- Управление оборудованием
- Управление персоналом
- Коллективная деятельность
- Коммуникации

Web-сервисы позволяют воспользоваться преимуществами одной из самых многообещающих возможностей Интернета – переводом обычных видов деятельности в онлайновый режим. Большинство компаний практически полностью отказалась от процесса обработки бумажных бланков, которые приходилось отправлять на утверждение. Автоматизировав этот бизнес-процесс, они и их компании-партнеры смогли на 50% сократить время, необходимое на обработку запросов, исключив весьма непроизводительный этап ручной обработки бумажных документов. Было исключено большое количество канцелярских ошибок и снижено количество потерь документов из-за неправильного заполнения или адресации. Сотрудники получили больше времени на рассмотрение заявок и реализацию мер по повышению качества, что привело к повышению точности работы, ускорению документооборота и улучшению степени удовлетворенности клиентов. Поскольку многие из этих компаний входили в состав более крупных объединений, в том числе виртуальных, повышение эффективности сказалось очень быстро, и выгоды для бизнеса стали заметны спустя очень непродолжительное время.

Применение архитектуры WSA облегчает процесс интеграции разнородных систем, кроме того, клиенты получают дополнительные преимущества за счет более эффективного использования возможностей Интернета. Коэффициент окупаемости инвестиций улучшается вследствие увеличения продуктивности, обусловленной внедрением Web-сервисов как во внутренних сетях, так и в экстрасетях. Если говорить о продуктивности, то здесь основной выигрыш связан с объемом времени, которое сотрудники тратят на решение различных частных задач, связанных с расширением

бизнеса. Важным фактором являются и преимущества более эффективного управления и улучшения консолидации информации, ранее содержавшейся в нескольких различных базах данных (как внутренних, так и внешних).

Одновременно с продуктивностью повышается и эффективность работы – показатель, характеризующий использование активов и непосредственно влияющий на доходность. Эффективность отражается на коэффициенте окупаемости инвестиций за счет уменьшения затрат, снижения темпов их роста или их устранения. Другими словами, речь идет о том, чтобы делать больше при меньших затратах. Например, одна из правительственных организаций США, используя Интернет-систему электронного взаимодействия, смогла стандартизировать и автоматизировать ведение отчетной документации и требования к данным для компаний, деятельность которых она регулировала. В свою очередь, автоматизация повысила эффективность работы самой организации и подотчетных ей компаний на 20-30%.

Многие из указанных преимуществ могут быть достигнуты за короткий промежуток времени, особенно при решении внутрикорпоративных задач, однако потребуется более длительный период времени для того, чтобы обеспечить полномасштабную совместную деятельность в рамках всей экономической среды, окружающей компанию. Кроме того, использование Web-сервисов для комплексных приложений потребует значительного времени, поскольку в процесс будут вовлечены множество сторон и технологий.

3.3.2Технологические преимущества выбора архитектуры WSA

Традиционные IT-архитектуры часто является закрытыми, что затрудняет возможности взаимодействия и привязывает клиентов к сделанным ранее инвестициям. Web-сервисы позволят компаниям использовать прошлые инвестиции, получая от них дополнительную отдачу. С точки зрения пользователей значительных барьеров на пути этой новой технологии нет, поскольку архитектура WSA - это первый крупный этап развития IT-архитектуры, который не требует от сообщества пользователей и не предполагает полной замены существующих IT-ресурсов.

Цель технологии Web-сервисов состоит в том, чтобы преодолеть ограниченность современных IT-технологий, разорвав жесткие фиксированные связи между приложениями и компонентами приложений, заменив ее более гибкими слабыми связями. Модель со слабыми связями, лежащая в основе Web-сервисов, не зависит от того, применяется ли она для соединения компонентов в рамках организации или для организации комбинированного сервиса, использующего функции от многих компаний.

Архитектура WSA позволяет организациям в большей степени сконцентрироваться на бизнес-процессах, вместо того чтобы заниматься вопросами технологии, помогая IT-менеджерам и специалистам по планированию процессов автоматизировать целый ряд операций и выполнять их быстрее и с меньшим риском:

- Определение мероприятий, необходимых для реализации бизнес-процессов
- Принятие решения о том, какие из указанных функций должны поддерживаться внутренними IT-средствами компании, а какие могут быть выполнены внешними сервисами
- Оценка того, какие сервисы должны быть доступны извне для клиентов, партнеров и поставщиков
- Обеспечение беспрепятственной совместной работы всех функций при одновременном обеспечении гибкости, достаточной для смены поставщиков или замены отдельных функций
- Устранение зависимости от базовых технологий
- Использование ранее созданных или реализация новых функций без привязки к предшествующим инвестициям в IT-систему

4 Вопросы интеграции в контексте Web-технологий

Положительное влияние архитектуры WSA позволит компаниям преодолеть многие из традиционных барьеров, присущих современным IT-технологиям. Это даст им большую свободу действий в создании новых бизнес-моделей и новых методов взаимодействия с заинтересованными сторонами.

Web-сервисы сегодня – это распределенные сервисы, обрабатывающие XML-кодированные SOAP-сообщения, описанные с помощью языка описаний Web-сервисов (WSDL).Web-сервисы обеспечивают интероперабельность между программными компонентами, которые могут взаимодействовать между различными компаниями и могут размещаться на различных инфраструктурах. Это решает одну из наиболее критичных проблем, с которой сталкиваются пользователи и разработчики программного обеспечения. HTTP и SOAP обеспечивают коммуникации и обмен сообщениями, WSDL обеспечивает описание сервиса для поддержки интероперабельности между средствами разработки, он дополнят интероперабельность коммуникаций возможностью обмениваться описанием интерфейсов.

Архитектура WSA – это инициатива, направленная на реализацию в полном объеме преимуществ распределенных вычислений, которые допускают объединение различными способами новых и ранее созданных вычислительных средств с целью поддержки развивающихся требований. Акцент на повторное использование компонентов обуславливает потенциал для значительного повышения продуктивности разработки и вытекающей из этого факта способности компаний быстро реагировать на новые потребности своей деятельности.

Одной из фундаментальных характеристик Web-сервисов является регулярная, состоящая из многих частей, структура сообщений. Эта структура обеспечивает композицию новой функциональности. Новые элементы, поддерживающие новые функции, могут быть добавлены к сообщению таким образом, что обработка существующей функциональности не изменяется. Например, можно независимо добавить идентификаторы транзакций и номера последовательностей надежных сообщений. Эти два расширения не входят в конфликт друг с другом и совместимы с ранее существовавшей структурой сообщений.

Как было отмечено выше, сервис-ориентированная архитектура (СОА) необходима для интеграции и является ключевой причиной для успеха. Одна из основных причин

активного использования к Web-сервисов заключается в том, что Web-сервисы хорошо подходят для реализации сервис-ориентированной архитектуры. При использовании Web-сервисов для построения сервис-ориентированной архитектуры решения заключаются в коллекциях автономных сервисов, идентифицируемых URL, с интерфейсами, документированными WSDL, и обработки правильно определенных XML сообщений. COA – естественное дополнение к объектно-ориентированному (OO), процедурному и основанному на данных подходов к реализации решения. На самом деле при создании COA-системы, отдельные сервисы обычно строятся с использованием одной или более из этих технологий.

Сервис-ориентированная архитектура отличается от OO и процедурных систем в одном ключевом аспекте: *связывании*. Взаимодействие сервисов основывается на том, *какие* они предоставляют функции и *как* они их предоставляют. OO и процедурные системы связывают элементы друг с другом на основании типов или имен. В отличие от них, модель Web-сервисов не оперирует с понятием разделяемых типов, требующих общей реализации. Вместо этого сервисы взаимодействуют на основе контрактов (WSDL/BPEL4WS для обработки сообщений) и схем (WSDL и XSD для структуры сообщений). Это позволяет сервису описать структуру сообщений, которые он может посылать и/или получать, и ограничения на последовательности сообщений. В связи с этим технологи контрактов и схем, разработанные для использования в сервис-ориентированном проектировании обеспечивают больше гибкости, чем традиционные объектно-ориентированные интерфейсы. В частности, сервисы используют свойства такие, как неконкретность XML элементов (например, xsd:any), расширения схем и необязательные SOAP блоки заголовков.

Наконец, для поддержания целостности Web-сервис-ориентация предполагает, что передаваемое сообщение может не только быть повреждено, но и что оно может быть послано с целью навредить или вообще с непредсказуемой целью. Следовательно, сервис-ориентированные системы защищают себя, требуя, чтобы приложения подтверждали, что поставщику сообщения были предоставлены требуемые права. Сервис-ориентированные архитектуры, согласованные с понятием автономии сервисов, обычно полагаются на административно управляемые доверительные соглашения, чтобы не полагаться только на механизмы аутентификации, связанных с сообщением, что является обычным в классических Web-приложениях.

Web-сервисы вызвали столь большой энтузиазм в частности благодаря их способности объединять разнородные системы. Разработчики предложили много полностью функциональных решений с использованием основных возможностей

транспорта, передачи сообщений и описания. Однако, для того чтобы быть принятыми разработчиками, создающими более мощные интегрирующие решения, Web-сервисы должны предоставлять функциональность, предоставляющую тот же уровень гарантии обслуживания, что и более традиционные решения среднего уровня. Недостаточно простого обмена сообщениями. Приложения и сервисы размещаются на среднем уровне и системы, предоставляющие значимые функции более высокого уровня, такие, как безопасность, надежность и операции с транзакциями, на основе Web-сервисов, должны обеспечивать механизм для интероперабельности между этими функциями.

Ниже кратко описываются основные предлагаемые спецификации Web-сервисов. Объясняется их значение для поставщиков, их роль в более широкой архитектуре и как они соотносятся друг с другом.

4.1 ИС и сервисы

4.1.1Постановка задачи

Спецификации транспорта и сообщений позволяют Web-сервисам взаимодействовать с использованием сообщений. При этом для успешного взаимодействия, требуется определение формата сообщений, которые Web-сервис потребляет и производит – интерфейса Web-сервиса. Группа описания спецификаций позволяет Web-сервису выразить свои интерфейсы и возможности.

В дополнение к интероперабельности сообщений эти спецификации обеспечивают также интероперабельность средств разработки. Спецификации описаний обеспечивают стандартную модель, которая позволяет разным средствам разных разработчиков совместно поддерживать разработчиков. Тем же способом, которым Web-сервисы изолируют партнеров от выбора реализации и инфраструктуры, спецификации описаний изолируют партнеров от выбора средств разработки.

4.1.2Обзор и анализ

4.1.2.1WSDL

Язык описания Web-сервисов (WSDL) и XML-схема (XSD) являются основными спецификациями в этой группе. XML-схема позволяет разработчикам и поставщикам сервисов определять XML-типы для структур данных, например, заказ на поставку, и сообщения, например, сообщение CreatePO (создать заказ на поставку). WSDL позволяет Web-сервису документировать сообщения, которые он посылает и получает. Другими

словами, какие «действия» или «функции» выполняет сервис в терминах сообщений, которые он получает и посылает.

WSDL обеспечивает поддержку нескольких вариантов взаимодействия сообщений. Он поддерживает односторонние входные сообщения, не имеющие ответа, запрос/ответ и односторонние посылки с или без ответа. Два последних варианта позволяют сервису определять нужные ему сервисы.

4.1.2.2WS-Policy

WSDL и XSD определения часто не дают достаточной информации для вызова сервиса. WSDL и XSD определяют синтаксис интерфейса сервиса, но не выражают информацию о том, как сервис поставляет свой интерфейс или чего сервис ожидает от вызывающего. Например, требует ли сервис обеспечения безопасности или реализует ли транзакции?

WS-Policy позволяет сервису специфицировать, чего он ожидает от вызывающего и как он реализует свой интерфейс. WS-Policy необходима для достижения интероперабельности на более высоком уровне функционирования сервиса. Безопасность, транзакции, надежная передача сообщений и другие спецификации требуют конкретной схемы WS-Policy. Это позволяет сервисам описывать функциональные гарантии, которых они ожидают от вызывающего и предоставляют ему.

WS-Policy представляет базовую модель для определения выражений политики.

WS-Policy поддерживает грамматику для утверждений политики интеграции и позволяет конструировать более гибкие и полные множества политик.

WS-PolicyAttachment определяют, как ассоциировать множество политик с XML элементами. Кроме того, WS-PolicyAttachment дает базовое множество общих утверждений, которые могут быть использованы для достижения интероперабельности

4.1.3Выводы

Применение стандарта WSDL для описания интерфейсов сервисов при использовании архитектуры WSA позволяет:

- определять структуры передаваемых в ходе взаимодействия с Web-сервисом данных в формате W3C XML Schema;
- структурировать передаваемые и возвращаемые параметры Web-сервисов с помощью понятия сообщений;
- группировать операции Web-сервисов в рамках различных портов, доступ к которым может осуществляться по разным URL;

- описывать привязку к транспортному протоколу с возможностью расширения стандартных элементов дополнительной спецификой конкретного протокола.

При следовании архитектуре WSA для описания интерфейсов Web-сервисов необходимо применять WSDL, а для описания структур данных, являющихся входными и выходными параметрами операций Web-сервисов необходимо использовать стандартный формат XML Schema, рекомендуемый к применению в WS-I Basic Profile 1.0.

Для описания регламентов взаимодействия с Web-сервисами (дополнительная семантика операций Web-сервисов, последовательность вызова операций за один акт взаимодействия) следует использовать грамматику для утверждений политики интеграции WS-Policy.

4.2 Транспорт и сообщения

4.2.1Постановка задачи

Интероперабельность Web-сервисов обеспечивается общим набором транспортных и технологий и технологий передачи сообщений. Чтобы убедиться, что эти технологии эффективны на практике, IBM, Microsoft и другие создали Организацию по обеспечению интероперабельности Web-сервисов (WS-I). В настоящее время WS-I выпустила базовый «профиль», который формально документирует интероперабельные механизмы транспорта и передачи сообщений Web-сервисов.

Задача ставится аналогично задаче разработки общего протокола обмена сообщениями (2.2.1).

4.2.2Обзор и анализ

Как было показано в разделе 2.2.3, для пересылки сообщений следует применять протокол SOAP. Вместе с тем следует учесть, что базовая спецификация допускает различные варианты использования этого протокола, что в конечном итоге привело к недостаточно хорошей совместимости различных его реализаций. Для решения проблем совместимости была создана рекомендация WS-I Basic Profile.

Поскольку протокол SOAP в базовой спецификации не поддерживает необходимые для среды требования, такие как поддержка транзакций или конфиденциальность пересылки, то данные требования должны быть реализованы с использованием механизмов расширений SOAP.

4.2.2.1 Транспортный уровень - HTTP, HTTP/S, SMTP

Этот набор спецификаций определяет базовые механизмы коммуникаций для передачи символьных представлений данных между Web-сервисами.

HTTP, HTTP/S и простой почтовый транспортный протокол (SMTP) служат примерами из этой группы. Реализации Web-сервисов могут, кроме того, поддерживать другие транспортные протоколы, но они обязаны обеспечивать поддержку стандартных интероперабельных протоколов.

В качестве альтернативных транспортных протоколов можно рассмотреть HTTPR, BEEP.

Протокол HTTPR обеспечивает надёжную доставку сообщений между приложениями в Интернет, с учётом возможного присутствия ошибок в каналах связи или программном обеспечении, а так же предоставляет средства уведомления о доставке. HTTPR работает поверх протокола HTTP/1.1.

BEEP (Blocks Extensible Exchange Protocol) является надстройкой над транспортными протоколами (такими, как TCP), которая облегчает разработку и использование протоколов приложений. Существует спецификация, позволяющая обмениваться SOAP сообщениями в рамках протокола BEEP.

Протоколы BEEP и HTTPR не следует применять в контексте Web-технологий, поскольку стандартным транспортным протоколом, на которой в большинстве случаев рассчитаны сетевые фильтры и приложения, является HTTP(S). В то же время, возможности надёжной доставки могут быть обеспечены внедрением стандарта WS-Reliable Messaging, который предоставляет те же гарантии, что и HTTPR, при этом работая поверх произвольного транспортного протокола, в частности, HTTP.

4.2.2.2 Форматы сообщений – XSD, SOAP

Следующая группа спецификаций определяет интероперабельные механизмы для кодирования сообщений Web-сервисов для передачи. Передача перемежает байтовые блоки, передаваемые между сервисами. Это полезно, только если участники могут преобразовать байты в полезные структуры данных, которые обрабатываются приложениями.

Группа спецификации сообщений определяет, как правильно форматировать сообщения. Определения XML и XML-схемы дают механизмы для абстрактных соглашений о структуре сообщений (данных). SOAP определяет стандарт кодирования для представления XML сообщений в байтовую информацию, которой сервисы обмениваются на транспортном уровне.

4.2.2.3WS-адресация

WS-адресация обеспечивает интероперабельный транспортно-независимый подход к идентификации отправителей и адресатов сообщений. WS-адресация обеспечивает также более тонкий подход к идентификации специфических элементов внутри сервиса, посылающего или принимающего сообщение.

Сегодня большинство систем, использующих Web-сервисы, кодируют адресата сообщения с помощью URL, который помещается в HTTP транспорт. Адресат ответа определяется адресом возврата. Используя сегодняшний подход, источник и адресат информации не являются частью самого сообщения. Это может вызвать ряд проблем. Информация может быть потеряна, если транспортная связь прерывается (например, ответ занимает много времени и связь прерывается по превышению времени) или если сообщение пересылается посреднику.

WS-адресация предоставляет механизм для размещения информации об адресате, источнике и другой важной адресной информации непосредственно внутри сообщения Web-сервиса. WS-адресация выделяет адресную информацию из любой специфической транспортной модели.

Во многих сценариях сообщения не направляются непосредственно сервису, и адресная информация в сообщении может быть описана просто с помощью URL. Но на практике мы часто находим, что сообщения направляются специфическим элементам или ресурсам внутри сервиса. Например, сервис координации может координировать много разных задач. Координатор нуждается в связывании большинства входящих сообщений с конкретным экземпляром задачи, которым он управляет, а не с сервисом координации как таковым.

Для адресации сущностей, управляемых сервисом, WS-адресация обеспечивает простой, но достаточно мощный механизм, называемый ссылка на конечную точку, который обеспечивает структурный подход к кодированию такого типа адресации.

Комбинация возможности указания точного адреса и независимого от транспорта кодирования источника и адресата сообщения позволяет посылать сообщения Web-сервисов через диапазон транспортов, через посредников, и обеспечивает поддержку различных образцов коммуникации.

WS-адресация позволяет также отправителю указывать независящим от транспорта образом, куда должен пойти ответ. Ответ на сообщение не обязательно отправляется отправителю. Например, в HTTP без WS-адресации невозможно определить, что ответ должен быть отправлен куда-нибудь еще.

Эти улучшения модели сообщений позволяют Web-сервисам использоваться для поддержки бизнес-сценариев. Например, некоторые банковские задачи на некоторых шагах требуют человеческого вмешательства. Обычно имеется много активных экземпляров задачи в каждый момент времени. WS-адресация обеспечивает общий механизм для связывания входящих или исходящих сообщений с конкретной задачей. Механизм, используемый сервисом, прозрачен для тех, которые используют сервис с помощью ссылки на конечную точку.

4.2.2.4Надежный обмен сообщениями

Среда Интернет является негарантированной системой доставки сообщений. Основные подходы и методы, позволяющие осуществить в этих условиях надёжную доставку, хорошо известны. Например, многие операционные системы присваивают сообщениям уникальные идентификаторы или последовательные номера и используют повторную передачу, если сообщение потеряно.

При отсутствии решения, обеспечивающего надёжную доставку сообщений на транспортном уровне, разработчики Web-сервисов должны были бы встраивать функции обеспечения надежности в сами приложения. Поскольку существует множество различных вариантов решения этой задачи, в конечном итоге это привело бы к несовместимости сервисов, предоставляемых разными разработчиками.

4.2.2.4.1 WS-ReliableMessaging

WS-ReliableMessaging определяет механизмы, которые позволяют Web-сервисам гарантированно доставлять сообщения по ненадежной коммуникационной сети.

WS-ReliableMessaging гарантирует, что сервисы используют интероперабельные подходы и позволяет поставщикам средств периода исполнения облегчить разработку приложений, предоставляя сервисы, реализующие протоколы. Это значительно упрощает задачу разработки приложений, уменьшая множество условий проверки ошибок, которые требуется обрабатывать.

Наконец, промышленность имеет богатый набор средств, ориентированных на сообщения среднего уровня для надежной передачи и распределения сообщений. Каждая реализация использует соответствующие протоколы. Протоколы WS-ReliableMessaging позволяют различным операционным системам и системам среднего уровня надежно обмениваться сообщениями. Таким образом; поддерживается объединение различных инфраструктур в единую логически полную модель.

4.2.3Выводы

При применении протокола SOAP следует учитывать следующее:

- Поскольку возможны различные варианты реализации протокола, то для наилучшей совместимости с существующими и будущими решениями следует использовать реализацию, соответствующую рекомендациям WS-I Basic Profile.
- В качестве транспортных протоколов следует использовать HTTP(S) и SMTP.

Возможные расширения протокола (поддержка транзакций, безопасности, композиции сервисов, надёжной доставки) следует включать по необходимости.

4.3 Каноническая модель данных и язык спецификации схем данных

4.3.1Постановка задачи

Для интеграции информации, соответствующей различным моделям данных, необходима каноническая модель данных, которая была бы максимально удобна для решения рассматриваемых задач. Нераздельно от выбора модели данных рассматривается выбор языка спецификации схем данных для этой модели – описаний предметных областей интегрируемых приложений и структуры информации.

Выбор технологии Web-сервисов как механизма интеграции обуславливает выбор XML как средства представления данных при обмене ими между интегрируемыми системами. Следовательно, больший приоритет при выборе канонической модели получают модели данных, разработанные в контексте Web, данные которых стандартным образом представимы в XML [1].

4.3.2Обзор и анализ

Сеть Web может рассматриваться как универсальный интерфейс к документам, базам данных и интерактивным сервисам. Развитие Web оказало существенное влияние на способ интеграции данных из различных источников, мы рассмотрим только модели данных, разработанные в контексте Web, который сегодня служит единой платформой для обмена данными и сервисами.

4.3.2.1XML и XML Schema

В качестве первичной модели данных при обмене информацией в распределённой Web-среде следует рассматривать модель XML, W3C eXtensible Markup Language [1].

Интерпретация того, что такое XML, зависит от выбранного ракурса: в простейшем случае XML может рассматриваться как синтаксис документов, но в зависимости от контекста возможны другие интерпретации. Изначально, XML - это язык, описывающий иерархически-структурированные документы. W3C выбрал XML за основу всей архитектуры Web, что объясняет также, почему его называют наследником HTML.

В контексте БД и информационных систем, XML - ещё один способ структуризации данных, с дополнительным преимуществом – текстовым представлением. Следовательно, текстовая форма XML-документов может также быть использована для обмена данными через сообщения, вот почему сообщество EDI (Electronic Data Interchange – электронного обмена данными) видит XML как стандартный синтаксис сообщений. С точки зрения данных, с появлением XML ощущается фундаментальный сдвиг в способе управления данными: XML-схемы имеют гораздо менее строгие ограничения, чем БД, и допускают большую гибкость. Это приводит к совершенно новой области – управлению слабоструктурированными данными.

4.3.2.1.1 Назначение XML

XML – это технология интеграции. С точки зрения **распределённых информационных систем**, XML начинает играть роль канонической модели данных, используемой для интеграции гетерогенной информации. Это является результатом того что, с одной стороны, XML играет двоякую роль: документ и модель данных (XML Infoset), а с другой стороны, XML обладает гибкостью и возможностью представления слабоструктурированных данных.

С точки зрения приложений, XML играет две важные роли. В **Web-информационных системах** важное преимущество XML перед HTML – это разделение структур данных и их представления пользователю. Это позволяет содержать данные целостными, повторно использовать их, обеспечить более сложную обработку данных. XML – формат для представления многообразия типов данных, в частности документов, позволяющий выделить ценную информацию из текста, упростить поиск и повысить адекватность поисковых запросов.

В **электронном взаимодействии** XML играет роль формата обмена сообщениями при реализации распределённых процессов внутри и между компаниями. Он заменяет используемые ранее форматы электронного обмена данными (EDI) и, будучи Web-стандартом, решает некоторые проблемы гетерогенности (в основном на уровне синтаксиса).

XML часто именуют lingua franca (общим языком) электронной коммерции. Стандартизуя применение XML, компании смогут взаимодействовать всегда и со всеми, без затрат на работы по интеграции, без которых нельзя было обойтись ранее.

4.3.2.1.2 XML-технологии

«XML будет ASCII для Web – простой, существенный, но ничем не выделяющийся» (Tim Bray). XML важен не столько своими характеристиками, сколько тем, что он является основой для целой архитектуры Web, покрывающей почти все вообразимые аспекты обработки информации. Эта разрабатываемая под эгидой W3C архитектура включает такие XML-технологии, как средства преобразования данных и привязки представления к данным, гиперссылки, схемы, программные интерфейсы, языки запросов, описание метаданных.

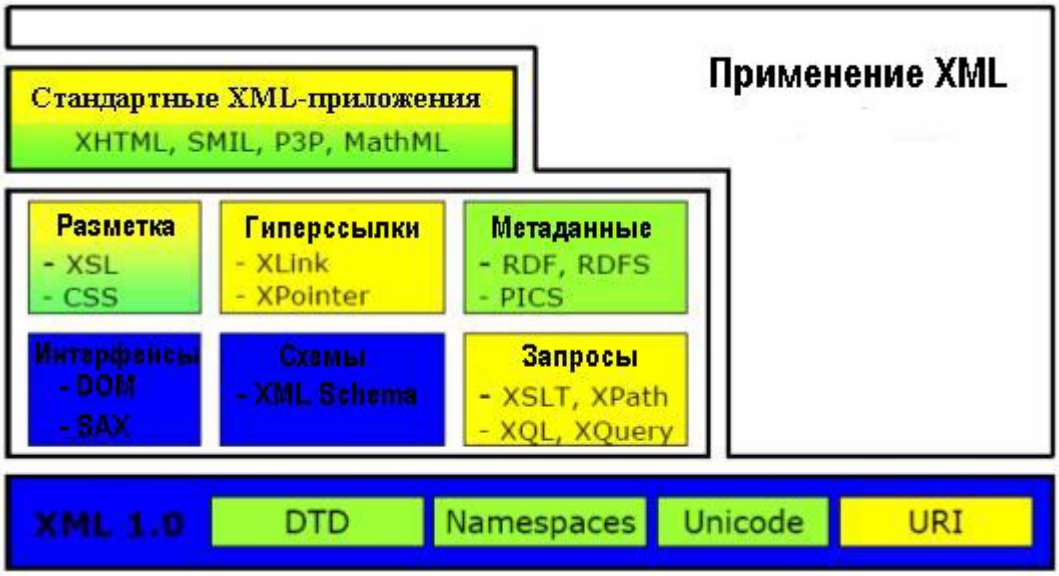


Рисунок 7. Области применения XML-технологий

4.3.2.1.3 XML Schema

Изначально XML-спецификация определяет язык DTD (Document Type Definition [1]), который используется для спецификации того, какие элементы и атрибуты допускаются в документе, и где они могут использоваться в документе по отношению друг к другу.

Тем не менее, по сравнению со схемами реляционных БД, DTD не определяет «типы», «сущности» в данных, а скорее определяет грамматику для документа.

Если же рассмотреть типичные ориентированные на представление данных XML-документы, то, очевидно, что они гораздо более походи на регулярно-структурированное представление сложных объектов, нежели гибко-структурированные документы (такие как XHTML). XML DTD до некоторой степени позволяют отразить эту структуру, но некоторые

аспекты DTD отразить не могут ввиду того, что они берут своё начало в мире обработки документов, а не обработки данных.

Отметим некоторые недостатки DTD. Во-первых, DTD имеют не-XML синтаксис. Это не столь важный недостаток, но он требует отдельных механизмов для манипуляции DTD в случае организации их хранилища, или же отображения DTD в XML-синтаксис, что делалось в ряде случаев.

Второй важный недостаток – это отсутствие поддержки модульности. Это особенно важно для сложных документов, и при взгляде на реальные DTD можно заметить широкое использование XML-сущностей с целью модуляризации. Однако, использовать этих сущностей опасно: поскольку они просто подставляются текстом, никогда нельзя быть уверенным в том, что тип используемых сущностей удовлетворяет требованиям. В мире объектной ориентации этот вопрос решается с помощью механизма наследования, и подобный механизм должен существовать и для XML.

В-третьих, DTD имеет очень ограниченный набор типов данных. По сравнению с моделями данных реляционных БД, это практически ничего.

Наконец, использование моделей содержимого для построения сложных типов, с одной стороны, даёт существенную гибкость и элегантность, но с другой стороны есть ряд ограничений типа, которые не могут быть выражены с помощью моделей содержимого, например, ограничения мощности или целостности ссылок.

Все эти недостатки DTD обусловили разработку нового стандарта для типизации XML-документов, а именно стандарта **XML Schema**, который тесно связан с объектно-ориентированной парадигмой (и моделью данных БД), то есть содержит много элементов из реляционной и объектной моделей, в частности ODMG и SQL99, но остаётся совместимым с конструкциями модели XML DTD. Кроме того, XML-схемы имеют XML-синтаксис.

Таким образом, XML Schema объединяет типичную парадигму объектно-ориентированного моделирования с конструкциями DTD в XML-синтаксисе.

4.3.2.2Semantic Web (RDF, RDFS, OWL)

“Semantic Web – это расширение Web, в котором информации придаётся определённая семантика, позволяя людям и машинам работать вместе“ – примерно такое определение дают своим работам члены W3C Semantic Web Activity. Целью этого проекта является внедрение в Web таких технологий, которые позволят существенно повысить уровень интеграции информации, обеспечить развитую машинную обработку данных. Semantic Web позиционируется как логическое продолжение развития Web – от

гипертекстовых страниц к XML-данным, а от XML – к смысловому содержанию и объединению разбросанной в Web информации.

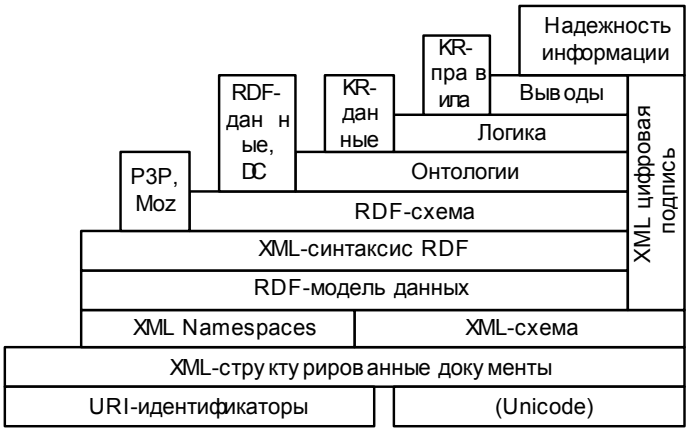


Рисунок 8. Общая архитектура Semantic Web

Модель данных Semantic Web, Resource Description Framework (RDF) [2,21], была специально спроектирована для интеграции распределённых метаданных в Web. RDF является представителем семейства ER-моделей данных: позволяет описывать объекты, или «ресурсы», указывая их «свойства» и значения свойств.

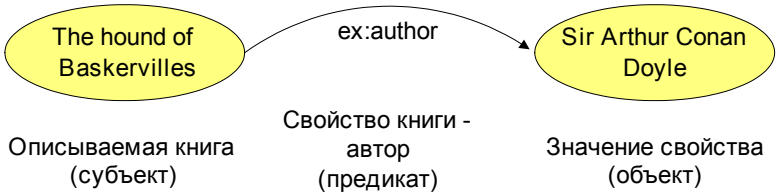


Рисунок 9. Пример описания свойства объекта

Ресурсы можно идентифицировать URI-идентификаторами [5], аналогично тому, как Web-страницы идентифицируются URL. Каждое свойство ресурса также указывается URI-идентификатором. Использование URI-идентификаторов вместо коротких имён – следствие «распределённости» информации, с которой мы сталкиваемся в Web. Другое следствие – это «децентрализация» информации: при описании некоторого ресурса в RDF-документе (скажем, человека) не обязательно описывать значения всякого его свойства (например, организацию, в которой он работает) – вместо этого можно указать это значение ссылкой по URI, аналогично тому, как в HTML мы указываем ссылки на другие Web-страницы. RDF – это синтез объектной ориентации, Web и XML, и позиционируется как рекомендуемая W3C модель данных для информационных источников Web.

RDF/XML-синтаксис [22] предоставляет возможность записи RDF-данных в форме XML для обмена информацией. RDF/XML – это способ смотреть на объектную информацию как на XML, и способ смотреть на XML-данные как на описание объектной информации нежели представление XML Infoset. Недостаток XML-схемы – это её техничность, семантическая бедность, которая обусловлена как раз тем, что XML-схема рассматривает XML-данные как XML Infoset, тогда как люди широко используют XML для представления объектной информации (ведь именно объектный подход наиболее близок к нашему видению вещей в реальном мире). RDF/XML позволяет избавиться от этого недостатка.

Язык RDF Schema (RDFS) [3] позволяет описывать на RDF словари классов и свойств. В XML Schema не хватает возможностей концептуального моделирования, а также средств интеграции независимых приложений: семантической интероперабельности. Эти вопросы решаются в RDF Schema. Описание обменных схем на RDFS и отображение схем на стандартизованные профили описания метаданных позволяет обеспечить существенно большую интероперабельность приложений. RDFS позволяет минимально смоделировать предметную область в терминах классов и их свойств, обеспечивая уровень концептуального моделирования, отчасти занимая нишу UML [17].

RDFS служит базой для более сложного языка описания «онтологий» предметных областей, Web Ontology Language (OWL) [24], который позволяет определить более сложные ограничения на применение классов и свойств, структуру данных. OWL имеет несколько уровней сложности языка, от простых расширений до ориентации на системы логики.

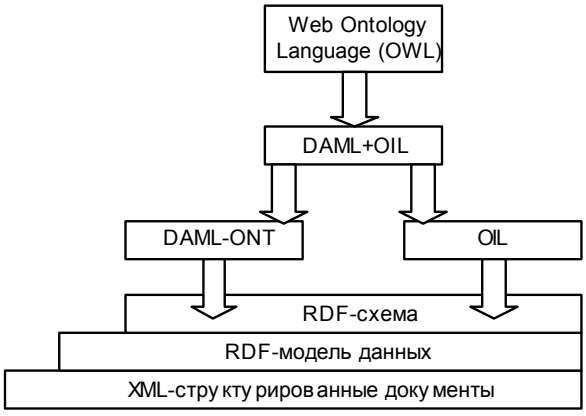


Рисунок 10. Иерархия языков описания объектной информации

Описание схем на RDFS и OWL можно рассматривать как более абстрактный концептуальный уровень описания XML-данных, тогда как XML Schema – более техническое описание конкретно формата XML-документов. RDFS/OWL рядом с XML Schema играют примерно ту же роль, что и ER-моделирование при разработке схемы реляционной БД.

4.3.3Выводы

Выбор технологии Web-сервисов как механизма интеграции обуславливает выбор XML как средства представления данных при обмене ими между интегрируемыми системами. Следовательно, больший приоритет при выборе канонической модели получают те модели, данные которых стандартным образом представимы в XML. Среди них выбор стоит между моделями данных XML (XML Infoset и XML-схемы) и Semantic Web (RDF, RDFS и OWL).

Semantic Web обладает более семантически богатой моделью данных (RDF) и средствами спецификации схем (RDFS, OWL), которые существенно лучше адаптированы к интеграции распределённых данных, обеспечению интероперабельности систем. Но внедрение технологий Semantic Web пока имеет существенный барьер ввиду их новизны и малой распространённости.

В связи с этим на данном этапе мы ограничиваемся распространённым и зарекомендовавшим себя «минимальным» вариантом канонической модели данных – XML. Для описания схем XML-документов из различных языков, описывающих схемы XML-документов (XML DTD, W3C XML Schema, RELAX), рекомендуется к использованию язык W3C XML Schema, поскольку этот язык стандартизован W3C и получил наибольшее распространение.

Однако в разделе «Руководство по стилю XML-схем» мы предлагаем рекомендации по стилю, обеспечивающему совместимость XML-схем с синтаксисом RDF/XML, то есть совместимость с технологиями Semantic Web. Эти рекомендации существенны как способ поддержки этих последних инициатив W3C, которые могут быть адаптированы в стандартах СЭВ и ИС ЭАР на следующих этапах.

Язык XML Schema небогат семантически, и содержит массу технических деталей представления данных, что требует дополнения его концептуальными моделями этих данных. Поскольку внедрение Semantic Web-технологий мы пока не рассматриваем, отличным выбором для описания таких концептуальных моделей можно считать язык моделирования UML. В разделе «Руководство по процессу разработки XML-схем» мы рекомендуем использовать XML Schema совместно с UML и при разработке схемы.

Перед описанием технических деталей представления информации в XML рекомендуется специфицировать её концептуальную модель в рамках объектно-ориентированной парадигмы с помощью UML-диаграммы классов. То есть, несмотря на то, что XML и XML Schema определяют каноническую модель данных при интеграции, при разработке схем рекомендуется всё же первичной считать объектно-ориентированную модель данных, как семантически более богатую.

4.4 Регистрация сервисов и метаинформации

4.4.1 Постановка задачи

В контексте Web-технологий задача регистрации сервисов и услуг аналогична той же задаче в общем контексте, за исключением того, что рассматриваемые стандарты должны поддерживать технологии Web-сервисов.

Существуют две основные задачи, связанные с регистрацией сервисов и метаинформации:

- поиск сервисов по различным критериям (название, поставщик сервиса, принадлежность рубрикам какого-либо классификатора).
- доступ к технической информации, достаточной для вызова сервисов.

Для описания интерфейсов и особенностей сервисов применяются такие технологии как XML, XSD, WSDL и WS-Policy.

4.4.2 Обзор и анализ

4.4.2.1 UDDI

При внедрении стандарта UDDI для регистрации сервисов следует учитывать, что UDDI реестр позволяет неоднозначно регистрировать сервисы и метаинформацию. Данные неоднозначности описаны в рамках спецификации WS-I:

- UDDI реестр позволяет задавать как непосредственный адрес сервиса, так и механизм получения адреса, основанный на промежуточном Web-сервисе. Второй механизм несовместим с WSDL, поэтому при регистрации следует указывать непосредственный адрес.
- Существует несколько возможностей хранения метаинформации, необходимой для вызова сервиса, в реестре. WS-I специфицирует, что для этого следует применять WSDL описание, и описывает технические детали, каким образом эта информация должна быть указана.

Ниже перечислены некоторые требования, которые eCo Framework (проект, разрабатывающий инфраструктуру для взаимодействия приложений, основанных на XML) рекомендует для UDDI реестра:

- Должен быть, как минимум, набор классификаторов, специфицирующих смысл общих терминов, таких как “город”. Кроме этого, должны быть подобные классификаторы, специфичные для предметных областей.
- Должен быть указан набор условий и ограничений, необходимых для вызова сервиса. Например, такие условия могут включать время суток, в которое доступен данный сервис.
- Описание сервиса должно указывать, что является результатом работы сервиса.

4.4.2.2WS-MetadataExchange

Для описания сервисов используются такие технологии WSDL, WS-Policy, и XSD. В широком смысле мы называем информацию о сервисе метаинформацией. Спецификация WS-MetadataExchange позволяет сервису предоставлять информацию другим посредством интерфейса Web-сервиса. Имея только ссылку на Web-сервис, потенциальный пользователь может получить доступ к множеству WSDL/SOAP-операций для получения метаданных, описывающих сервис. Клиенты могут использовать WS-MetadataExchange в процессе проектирования при построении приложений или во время исполнения.

4.4.3Выводы

При использовании UDDI реестра необходимо следовать рекомендациям WS-I:

- При регистрации сервиса следует указывать его непосредственный адрес. UDDI позволяет использовать вместо этого адрес сервиса разрешения; эту возможность следует избегать.
- Для описания технической информации, необходимой для вызова сервиса, следует применять стандарт WSDL. WSDL описание должно быть ассоциировано с сервисом с использованием рекомендованного в спецификации UDDI механизма.

4.5 Безопасность

4.5.1Постановка задачи

Семейство спецификаций обеспечения безопасности критично для взаимодействия Web-сервисов в условиях незащищенных каналов связи, в частности, при передаче данных через Интернет. Данные спецификации поддерживают аутентификацию отправителя сообщений, их целостность и конфиденциальность. Они также поддерживают корпоративную безопасность между различными организациями.

4.5.2Обзор и анализ

В настоящее время разрабатываются две группы стандартов, направленных на обеспечение безопасного взаимодействия Web-сервисов. Это стандарты WS-Security организации OASIS и группа стандартов Liberty. Эти наборы стандартов имеют схожие задачи. Средства их решения также примерно одинаковы. Стандарты Liberty идут несколько впереди разработок OASIS, при этом в стандарты OASIS включаются идеи и решения Liberty.

Наиболее существенными для рассматриваемого класса систем являются: WS-Security, WS-Trust, WS-SecureConversation и WS-Authorization для взаимодействия Web-сервисов в рамках одной ИС, а также WS-Policy и WS-Federation для организации взаимодействия между различными ИС.

В связи с большим выбором вариантов использования этих стандартов необходимо ввести профиль использования, т.е. соглашение об использовании стандарта. Организация WS-I разрабатывает профили для web-стандартов: Basic Profile Version 1.0 и Basic Security Profile Version 1.0 (последний сейчас в статусе драфта). В OASIS также разрабатываются профили использования WS-Security, например «SOAP Message Security: Minimalist Profile (MProf)», но эти работы или свернуты и перенесены в WS-I, или развиваются слишком медленно, поскольку последняя версия Mprof датируется мартом 2003 года.

В рамках OASIS разрабатываются также языки для представления информации, необходимой для функционирования сервисов безопасности: SAML и XACML.

SAML предназначен для безопасного кодирования аутентификационной и авторизационной информации в теле SOAP-запроса, что позволяет уменьшить необходимое для операций аутентификации и авторизации число сетевых сообщений, которыми должны обменяться сервисы инфраструктуры безопасности, а также реализовать сервис единого входа.

XACML – это стандарт OASIS, специфицирующий язык описания политики доступа к ресурсам. Его сильной стороной является гибкость, расширяемость и универсальность. Спецификация языка не делает предположений относительно деталей реализации сервиса контроля доступа. Для эффективного применения языка необходимо разработать профиль, описывающий схемы идентификации субъектов, объектов доступа и операций над ними.

Субъектами являются: внутренние пользователи информационных систем (ИС), внешние пользователи ИС, Web-сервисы, сервера инфраструктуры безопасности. В настоящее время практически все средства идентификации субъектов можно разделить на следующие категории:

- средства на базе асимметричной криптографии;
- средства, основанные на знании субъектом определенной информации или обладании им определенной сущностью;
- средства, основанные на неотъемлемых характеристиках субъекта.

Системы первой категории сложнее в реализации, но предпочтительнее, поскольку позволяют реализовать дополнительные сервисы безопасности, такие как non-repudiation (невозможность отрицания субъектом своих действий).

Наиболее распространены следующие системы идентификации:

В первой категории: PKI (Public Key Infrastructure) на основе X.509, SPKI (Simple Public Key Infrastructure), PGP (Pretty Good Privacy). Во второй категории: пара имя/пароль, биометрическая идентификация (фактически: фиксированный пароль на основе биометрических данных).

Наиболее широко используемой реализацией PKI является стандарт X.509. Преимуществами этого стандарта являются:

- широкая поддержка технологии и стандарта X.509 большим количеством производителей;
- наличие широко распространенной инфраструктуры;
- возможность универсальной идентификации любых субъектов.

Технология SPKI является упрощенной версией PKI, разрабатывалась рабочей группой IETF, но в 1999 году работы по стандартизации были прекращены, не в последнюю очередь из-за непринятия предложений крупными производителями. В настоящий момент является технологией ограниченного назначения, применяемой в распределенных, но закрытых системах.

PGP – технология на основе ассиметричной криптографии, используемая в основном для проверки целостности сообщений. Недостатком технологии является трудность идентификации владельца публичного ключа.

В случае невозможности использования сертификатов X.509 для идентификации субъекта допускается использовать комбинацию имя/пароль. Такой метод идентификации может использоваться только для идентификации внешних пользователей системы, которые не могут выполнять действий, имеющих юридические последствия.

4.5.2.1WS-Security

WS-Security – основной строительный блок для безопасных Web-сервисов. Сегодня большинство распределенных веб-сервисов полагаются на транспортный уровень поддержки функций безопасности. Примерами являются HTTP/S и BASIC-Auth аутентификация. Эти подходы к безопасности обеспечивают минимум, необходимый для безопасных коммуникаций. Уровень функции, который они обеспечивают, однако, существенно ниже того, что предоставляют существующие среды среднего. Два примера иллюстрируют недостатки HTTP/S и BASIC-Auth.

- A посылает сообщение B. B частично обрабатывает сообщение и пересылает его сервису C. HTTP/S допускает аутентификацию, конфиденциальность и целостность между A-B и B-C. Однако, C и A не могут аутентифицировать друг друга и скрыть информацию от B.
- Для A, B и C для аутентификации используется BASIC-Auth. Они должны иметь одного и того же общего дублируемого пользователя и информацию о пароле. Во многих сценариях это недопустимо.

WS-Security решает эти проблемы. Он поддерживает:

- Подписанные, кодированные маркеры. A может генерировать маркеры, которые C может проверять как идущие от A. B не может подделать маркер.
- A может подписать избранные элементы или целое сообщение. Это позволяет B и C подтвердить, что сообщение не изменилось после того, как A послал его.
- A может оставить открытым или запечатать сообщение или избранные элементы. Это гарантирует, что только сервис, которому предназначались элементы, может использовать эту информацию, т.е. предотвращает B от просмотра информации, предназначенной для C и наоборот.

WS-Security использует существующие модели безопасности (Kerberos, X509, и т.д.). Спецификации конкретно определяют, как использовать существующие модели интероперабельным образом.

4.5.2.2WS-Trust

Безопасность основывается на предопределенных отношениях доверия. Kerberos работает потому, что участники «доверяют» центру распределения ключей Kerberos. PKI работает потому, что участники доверяют центру проверки сертификации маршрутов. WS-Trust определяет расширяемую модель для установления и проверки отношений доверия.

Ключевая концепция WS-Trust – это Служба Компонент Безопасности - Security Token Service (STS). STS – это специальный Web-сервис, который выдает, осуществляет обмен и проверяет компоненты безопасности. WS-Trust позволяет Web-сервисам устанавливать и договариваться о том, каким серверам они «доверяют» и полагаться на эти серверы.

STS имеет широкое применение, поскольку может использоваться для выдачи компонент безопасности, которые используют широкий диапазоне утверждений. Во многих случаях STS будет использоваться для выдачи одних и тех же утверждений, но в различных форматах. Например, STS может выдать Kerberos компоненту, утверждающую, что владельцем ключа является фирма XYZ и это может быть сделано на основе сертификата X.509, выданного Сообществом Сертификатов, которому доверяют. Это позволяет участвовать организациям, использующим другие технологии безопасности.

4.5.2.3WS-SecureConversation

В некоторых сценариях Web-сервисов происходит только нерегулярный обмен несколькими короткими сообщениями. Эта модель вполне поддерживается WS-Security. В других сценариях происходит длительный обмен многими сообщениями между Web-сервисами. WS-Security также поддерживает эту модель, но это решение не оптимально.

В таких сценариях есть две причины неоптимальности использования:

- Повторное использование сложных вычислений в криптографических операциях таких, как проверка публичных ключей.
- Отправка и прием многих сообщений с использованием одних и тех же криптографических ключей, что дает больше информации для осуществления грубых атак, пытающихся «взломать код».

По этим причинам протоколы типа HTTP/S используют обычные ключи для выполнения переговоров, которые определяют ключи, специфичные для данной сессии. Такой обмен ключом обеспечивает более эффективную реализацию безопасности и одновременно уменьшает объем информации, закодированной специфичным набором ключей.

WS-SecureConversation обеспечивает аналогичную поддержку для WS-Security. Участники часто используют WS-Security с «публичными» ключами для начала «контакта» или «сессии» и используют WS-SecureConversation для установления соглашения о ключах, специфичных для данной сессии, для подписывания и кодирования информации.

4.5.2.4WS-Federation

WS-Federation позволяет нескольким организациям установить единую виртуальную область безопасности. Например, такое сообщество может быть образовано цепочкой агентство путешествий, авиакомпания и отель. Конечный пользователь, который «заходит» к любому из членов сообщества по существу заходит ко всем его членам. WS-Federation определяет несколько моделей предоставления безопасности сообщества посредством протоколов между топологиями WS-Trust и WS-SecureConversation.

Кроме того, пользователи часто имеют особые «свойства», когда они вступают в переговоры с предприятиями. Примером является предпочтение сидеть у окна, у прохода или в середине. WS-Federation позволяет участникам устанавливать общее пространство свойств. Это позволяет каждому участнику иметь безопасный управляемый доступ к информации каждого члена о конечных пользователях.

Свойства и информация о личностях может быть защищена по причине конфиденциальности или потому, что эта информация дает преимущество в конкуренции. Для поддержки этих требований WS-Federation предлагает *модель псевдонимов*. Пользователи, которые аутентифицировались в агентстве путешествий, имеют генерированные «псевдонимы» при взаимодействии с авиакомпанией или отелем. Это защищает частную информацию конечного пользователя и конкурентные преимущества, которые агентство путешествий могло бы получить, зная свойства пользователя.

4.5.3Выводы

Учитывая большую поддержку OASIS, предлагается в качестве базового стека стандартов выбрать WS-Security от OASIS. В областях, где работа еще далека от завершения, следует ориентироваться на результаты Liberty. С целью упрощения стандартов и обеспечения интероперабельности предлагается использовать профили организации WS-I: Basic Profile Version 1.0 и Basic Security Profile Version 1.0. В качестве

языка описания политик доступа следует использовать XACML. В качестве системы идентификации предлагается использование PKI на основе сертификатов X.509 v3.

В целом, стандарты безопасности web-сервисов, находятся на начальном этапе развития. Настоящее исследование является лишь первым этапом, позволяющим определить круг задач, требующих решения, и группы стандартов и технологий в комплексе позволяющие эти задачи решить. Дальнейшее исследование стандартов, их перевод, а также создание их профилей, необходимых для рассматриваемой предметной области, будет выполнено на последующих этапах.

4.6 Транзакции

4.6.1Постановка задачи

В случае слабо-связанной модели, поддерживаемой Web-сервисами, становится ясно, что семантика традиционных ACID-транзакций не подходит для многих Web-приложений. Транзакции, основанные на Web-сервисах, отличаются от традиционных транзакций тем, что они выполняются длительное время, требуют, чтобы необходимость фиксации транзакции определялась во время исполнения, и чтобы уровень изолированности был снижен.

4.6.2Обзор и анализ

4.6.2.1OASIS Business Transactions Protocol (BTP)

BTP был первой попыткой создать XML-стандарт для транзакций типа b2b. Он разработан для поддержки приложений, которые несопоставимы по времени выполнения, расположению и управлению, и тем самым требуют поддержки транзакций на уровне, выходящем за пределы классических ACID-транзакций. Это протокол управления бизнес-процессами, представленными слабосвязанными программными сервисами, для получения устойчивых (непротиворечивых) результатов от участвующих бизнес-партнеров.

BTP вводит два типа *расширенных транзакций*, оба использующих двухфазовый протокол завершения:

- *Атом*: атом – это типичный способ выделения области действия "транзакционных" действий, выполняемых Web-сервисами. Результат атома гарантированно будет атомарным, поэтому все вовлеченные *участники* (действующие от лица их ассоциированных Web-сервисов) получат один тот же результат, и он либо подтвердит выполненные действия, либо отменит их. Несмотря на то, что на

первый взгляд может показаться, что атомы BTP эквивалентны атомарным транзакциями, на самом деле эквивалентности там нет. BTP не рассматривает интероперабельность с существующими системами поддержки транзакций как важный фактор. Семантика атома (изолированность, устойчивость и т.д.) не определены так точно, как для атомарных транзакций.

- *Звено*: этот тип транзакций был введен для того, чтобы ослабить атомарность, принять во внимание то, что выбор между подтверждением или отменой произведенных действий базируется на бизнес-правилах высокого уровня. Атомы являются типичными участниками в звеньях, но в отличие от атома звено может давать различные результаты его участникам для того, чтобы некоторые из них могли подтвердить их действия, в то время как остальные - откатить. По существу, двухфазовый протокол для звена параметризован, чтобы позволить пользователю конкретно указать каких участников (атомов или автономных участников) готовить и каких откатывать. Основанием идеи звеньев служит то, что они лучше позволяют смоделировать долго выполняемые бизнес процессы, где сервисы включаются в атомы, представляющие конкретные единицы работы, и по мере того как бизнес-процесс выполняется, он может попасть в условия, которые позволяет ему отменить или приготовить эти элементы с предупреждением. Причем это может быть за много часов до того, как звено примет конечное решение и определит так называемое «множество подтверждения»: набор участников, которых нужно подтвердить для того, чтобы успешно завершить бизнес-процесс. Как только множество подтверждения определено, звено сворачивается в атом – все члены множества подтверждения видят один и тот же результат.

Подход BTP к Web-сервисам также отличается от того, который можно ожидать. С самого начала, технический комитет решил, что BTP будет полезен и за пределами Web-сервисов. Как таковой, BTP не ориентирован на Web-сервисы; он не использует архитектуру Web-сервисов, не содержит привязки в виде WSDL или протокола взаимодействия. Это означает, что вместо того, чтобы определять требования к конкретному механизму BTP определяет полный набор сервисов в транзакционном протоколе.

К сожалению, отображение BTP на конкретную среду распределенных приложений, такую как, например, Web-сервисы, может означать, что определенные части набора не нужны; также может оказаться, что функциональность среды может даже пересекаться с транзакционным протоколом.

Интересно, что всё, что дает BTP, это набор XML-сообщений, необходимый для обслуживания протокола. Метод обмена этими сообщениями между различными участниками может определяться средой (например, email или HTTP), типом бизнес-отношений или другими бизнес-факторами. Спецификация определяет привязку к SOAP поверх HTTP, но это не является принудительным. Не существует базового понятия интероперабельности для BTP (для поведения протокола). Это просто стандартизация содержания сообщений и последовательности сообщений.

4.6.2.2WS-Coordination Framework и протоколы распределенных транзакций WS-Transaction

В отличие от BTP, который привязывает координирования к транзакциям, протокол Web Services Transactions использует отдельный протокол, предназначенный исключительно на вычисление/обработку результата: Web Services Coordination. Фундаментальная идея в основе WS-Coordination заключается в том, что существует общая необходимость в координирующей инфраструктуре в среде Web-сервисов. Спецификация WS-Coordination определяет координирующую среду, которая позволяет подключить различные координирующие протоколы для координации работы между клиентами, сервисами и участниками.

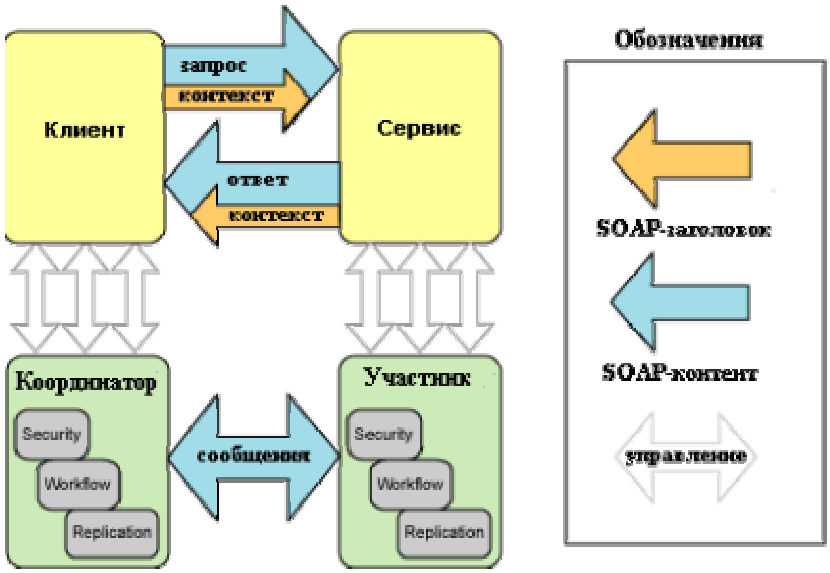


Рисунок 11. Обмен сообщениями в координирующей среде

Оба протокола WS-Coordination и WS-Transaction предназначены исключительно для среды Web-сервисов и, таким образом, используют существующие и возникающие стандарты, такие как WSDL, WS-Addressing, Web Services Security и WS-Policy. Это

фокусирует WS-Coordination и WS-Transaction на среду Web-сервисов, упрощает спецификации и полагает их компонентами в архитектуре Web-сервисов.

Координация – это действия одного агента (*координатора*) распространяющего информацию некоторому количеству участников для того, чтобы гарантировать, что все участники получат специальное сообщение. Координатор так же может быть участником, создающим древовидную структуру подчиненных координаторов (sub-координаторов или peer-координаторов), которые взаимодействуют для дальнейшей передачи результата. В отличие от BTP, посреднические взаимодействия являются существенной частью моделей WS-Coordination и WS- Transaction.

Информация о состоянии поступает неявно (прозрачно для приложения), когда нормальные сообщения посылаются участникам. Эта информация специфична для типа выполняемой координации, например, таких как определение координатора(ов), других участников действия, информация для восстановления в случае сбоя и т.д. Более того, может оказаться так, что дополнительно требуемая специфичная в контексте приложения информация (например, дополнительная информация в заголовках SOAP) передается тем участникам или сервисам, которым она нужна.

WS-Coordination определяет общую координирующую среду, которая может поддерживать произвольные протоколы координации. Протоколы координации расширяемы и на уровне координаторов, и на уровне контекста. Например, координатор, выполняющий трехфазовый протокол фиксации, может легко быть подключен в реализациюWS-C и базовый контекст WS-C может быть при необходимости расширен. Данная среда полезна для распространения набора типов контекста, включающих безопасность, управление потоками работ и репликацию.

Спецификация WS-Coordination работает в терминах *действий*, распределенных единиц работы, вовлекающих в процесс одного или более участников (которые могут быть сервисами, компонентами или просто объектами). На этом уровне, действие минимальным образом определено, просто создается, запускается и, затем, завершается.

Для чего бы не использовался координирующий протокол, всегда представлены одинаковые требования:

- Создание экземпляра (или активация) нового координатора для конкретного координирующего протокола, для отдельного экземпляра приложения.
- Регистрация участников у координатора, чтобы они могли получать сообщения координирующего протокола во время (некоторой части) времени жизни приложения.

- Распространение контекстной информации между Web-сервисами, составляющими приложения.
- Объект для подведения координирующего протокола к завершению.

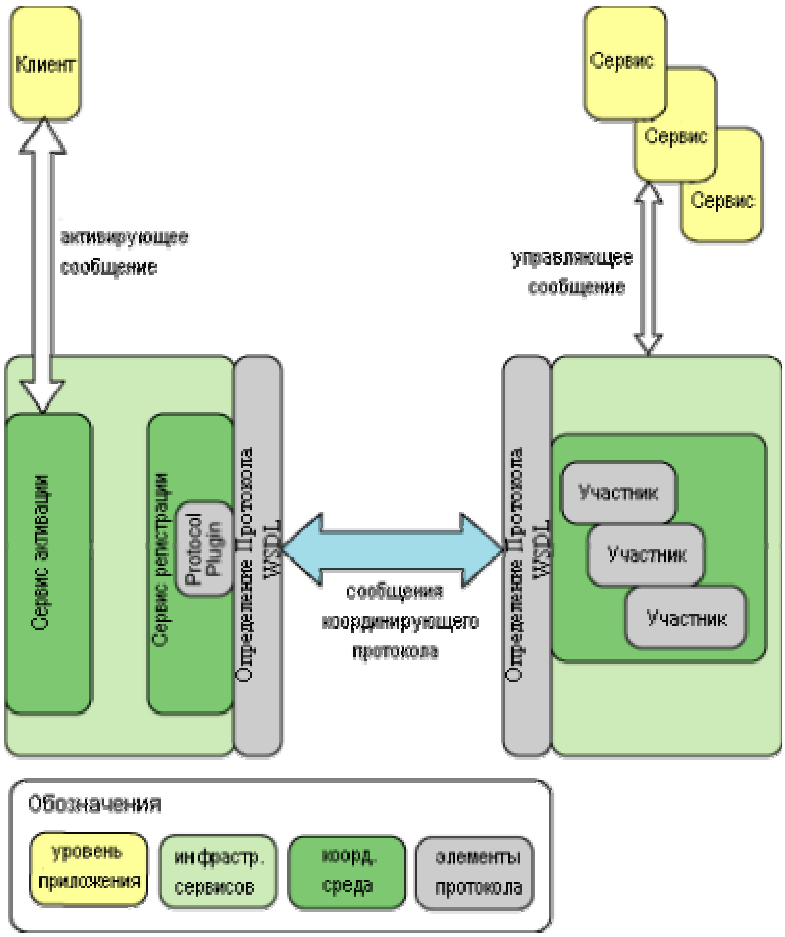


Рисунок 12. Взаимодействие в рамках координирующих протоколов

Среда WS-Coordination представляет *Сервис Активации,* который поддерживает создание координаторов для конкретных протоколов и их ассоциированных контекстов. Процесс вызова сервиса активации изображен происходящим асинхронно, так что спецификация определяет интерфейс и самого сервиса активации и сервиса вызова: сервис активации может сделать обратный вызов для передачи результатов активации, а именно контекста, идентифицирующего тип протокола и расположение координатора.

Как только сервисом активации созданы экземпляр координатора и соответствующий контекст, начинает работу *Сервис Регистрации*. Этот сервис позволяет участникам регистрироваться для получения сообщений протокола, ассоциированного с их конкретным координатором. Подобно сервису активации, сервис регистрации предполагает асинхронное взаимодействие и поэтому специфицирует WSDL и для сервиса регистрации, и для регистрирующегося клиента.

Спецификация WS-Transaction включается в WS-C и предлагает два общих шаблона завершения (конкретных координирующих протокола), каждый из которых поддерживает семантику конкретного типа взаимодействия:

- *Атомарные транзакции (AT)*: этот шаблон предназначен для удовлетворения существующих стандартов, которые имеют четко определенные понятия для атомарности (две фазы), изолированности (отсутствие аномалий «грязного» и «повторяющегося» чтений) и устойчивости (отсутствие потерь данных), другими словами, традиционной семантики ACID. Важная вещь, которую нужно помнить, рассматривая Web-сервисы, это то, что они предназначены для достижения интероперабельности так же, как и что они предназначены для Web. Раньше заставить две традиционные транзакционные системы взаимодействовать друг с другом можно было только ценой больших усилий; Web-сервисы здесь предлагают не имеющую себе равных поддержку интероперабельности. Традиционные транзакционные системы уже сформировали основу приложений enterprise-уровня, и так будет для эквивалента на Web-сервисах. Процессы b2b будут обычно прямо или косвенно использовать вспомогательные системы обработки транзакций и возможность связать существующие системы будет ключом к успеху транзакций на уровне Web-сервисов. В конце концов, атомарные транзакции полезны только для внутренних сред, где пользователь хочет объединить действия некоторого числа внутренних приложений. Например, поглощение одной компанией другой может вызвать необходимость комбинации приложений обеих.
- *Бизнес-процесс(Business Activity)*: этот шаблон предусматривает гибкие параметры транзакций и разработан специально для длительных по времени взаимодействий, где удерживание ресурсов невозможно или непрактично. В этой модели сервис получает запрос на выполнение работы (например, резервирования места на рейс), и если это действие выполняется так, что позже результат может быть возвращен в предыдущее состояние (“откат”), то сервис может информировать бизнес-процесс о результатах. Таким образом, если бизнес-процесс позже решает, что работу нужно отменить, он может проинформировать сервис. Как сервисы выполняют их работу, и какие механизмы компенсации используются, не определяется спецификацией WS-Tx: это определяется конкретной реализацией сервис-провайдера.

Как уже отмечалось, спецификации Web Services Coordination и Web Services Transactions твердо заняли своё место в архитектуре Web-сервисов. По существу, они

разработаны с возможностью при необходимости использовать другие спецификации Web-сервисов, такие как безопасность, надежные сообщения т.д. Однако, в отличие от BTP, эти другие требования четко разграничены в отдельные спецификации.

4.6.3 Выводы

Следующая таблица показывает резюме по различиям и сходствам WS-C/T и BTP.

Таблица 3. Резюме по различиям и сходствам WS-C/T и BTP

	WS-C/Tx	**BTP**
Координирующая система	WS-C	Отсутствует – привязка к двум фазам.
Транзакционная система	Отсутствует, но определенные протоколы покрывают типичные шаблоны (АТ и БП); другие будут добавлены в дальнейшем.	Общий протокол, без шаблонов. Статически определен.
Строгая атомарная модель	**Атомарная транзакция** – атомарные транзакции требуют строгих ограничений ACID. Конкретно – для интероreабельности с традиционными транзакционными системами.	**Атом** – атом обладает лишь свойством атомарности, остальные свойства определяются сервисом (и недоступны по протоколу). Использует открытый протокол; обеспечение интероперабельности является сложной проблемой.
Ослабленная модель	Бизнес-процесс позволяет использовать меняющийся список участников.	Звенья позволяют использовать меняющийся список участников. Требует от участников открытости приложению.
Области действия	Есть. БП управляет связями между областями действия. Разрешены вложенные области видимости.	Нет. Звено управляет связями внутри области действия.
Разные	Есть, определяются бизнес-	Есть, определяются звеном.

результаты для контекстов.	процессом.	
Гибкий список участников контекста.	Есть.	Есть, участники могут выходить из звена. (Выход из атома эквивалентен режиму «только чтение» в Атомарной Транзакции)
Поведение сервисов	Определяется протоколами	Определяется сервисами (не спецификацией BTP).
Разделение бизнес-логики и координатора	Четкое.	Смешанное. Протокол требует жесткой привязки бизнес-логики и координатора.
Ориентировано на Web-сервисы	Да	Нет. Требует большого количества дополнительной работ от спецификации/протокола.
Восстановление после сбоя	Оптимизированный протокол.	Оптимизация выполняется отдельно для каждой распределенной среды.

В реальной бизнес-среде существует больше деталей и особенностей. WS-Coordination и WS-Transaction разработаны для того, чтобы согласованно управлять всеми этими уровнями методом, основанным на стандартном наборе Web-сервисов и дополняющим его.

В дополнение к этому, благодаря тому, что семантика Атомарных Транзакций и их участников четко определена, пропадает неопределенность у пользователей и сервисов: они знают, какую семантику ожидать, так как это является требованием протокола. Так как в BTP по сути имеется только один тип транзакций (атомы – это подмножества звеньев), эта спецификация должна позволять гибкие реализации участников долго выполняемых взаимодействий, тем самым BTP не накладывает четкие ограничения на семантику участников. Решение о том, как сделать эту информацию доступной пользователям за пределами области действия протокола транзакции, предоставляется создателю сервисов/участников.

Бизнес-процесс по спецификации WS-Tx дает разработчикам сервисов свободу определения механизмов компенсации, наилучшим образом подходящих их сервисам (например, использование Атомарных Транзакций в случае необходимости), в то же время предоставляя простую модель пользователям этих сервисов. В добавок к этому, он

хорошо связывается с технологиями организации совместной работы Web-сервисов, такими как BPEL4WS.

Существенно также то, что поскольку WS-Tx использует WS-C, он представляет собой набор расширенных моделей транзакций, каждая из которых предназначена для конкретной области задач. Видно, что не все задачи могут быть решены с помощью существующих в WS-T протоколов, поэтому существует возможность расширения, добавления новых протоколов.

Модели бизнес-процесса, координации и управления транзакциями, предлагаемые спецификациями WS-Coordination и WS-Transaction, помогают пользователю оценить некоторые ключевые выгоды:

- Они основаны на архитектуре Web-сервисов. Web-сервисы позволяют приложениям, выполняющимся на разных, потенциально несовместимых платформах, интегрироваться и динамически и гибко взаимодействовать друг с другом. Используя эту базу, спецификации WS-Coordination и WS-Transaction способствуют ускорению процесса интеграции и реализации более гибких бизнес-процессов.
- Они расширяемы. WS-Coordination и WS-Transaction, в своей основе предоставляют обобщенную систему, определяющую как может быть реализован бизнес-процесс, но в то же время дающую значительную свободу для расширения и подгонки деталей реализации под конкретные бизнес-нужды. Это позволяет развивать программные средства как только появляются новые требования, что сокращает затраты на разработку.
- Они гибки. Система, предлагаемая спецификациями WS-Coordination и WS-Transaction, обеспечивает поддержку широкого спектра транзакционных бизнес-процессов.
- Они обеспечивают устойчивую и надежную работу. Спецификация WS-Transaction предоставляет средства наблюдения за надежностью каждой индивидуальной задачи, входящей в общий процесс.
- На уровне бизнеса спецификации WS-Coordination и WS-Transaction позволяют бизнес-процессам достигать своих целей путем, позволяющим им легко интегрировать свои бизнес-действия с теми партнерами или пользователями, которые были выбраны для использования, вне зависимости от их платформ разработки и выполнения.

4.7 Рабочие процессы

4.7.1Постановка задачи

Постановка задачи описания автоматизированных рабочих процессов аналогична соответствующей постановке из подраздела «2.7.1 Постановка задачи» раздела «2. Общие вопросы интеграции» данного документа.

4.7.2Обзор и анализ

На данный момент различными разработчиками предложен ряд языков для описания рабочих процессов. К основным стандартам можно отнести следующие:

- XPDL (Workflow Management Coalition);
- BPML (Business Process Management Initiative);
- WSFL (IBM);
- XLANG (Microsoft);
- BPEL4WS (IBM, Microsoft).

Все перечисленные спецификации имеют свои характерные особенности, определяющие преимущества в функциональности основанного на их базе решения.

4.7.2.1XML Process Definition Language (XPDL)

Спецификация XPDL, предложенная WfMC - Workflow Management Coalition, представляет собой формальную модель для описания рабочих процессов, относящихся к любым сферам деятельности. В соответствии с ней каждый поток работ разбивается на следующий набор взаимодействующих между собой компонентов:

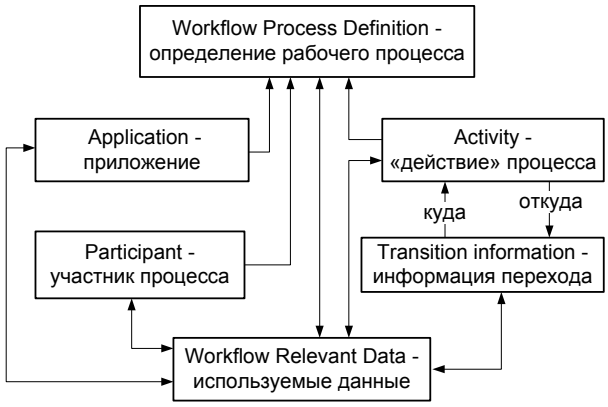


Рисунок 13. Метамодель XPDL-процесса

- *Workflow Process Definition* – представляет собой контекст выполняющегося процесса, и его данные могут быть доступны всем остальным компонентам.
- *Activity* – «действие» или «задание» процесса, представляющее собой этап, на котором происходит изменение содержания объектов процесса.
- *Transition information* – переходы между заданиями (могут быть условными и безусловными).
- *Workflow Relevant Data* – оперативные данные, доступные всем компонентам процесса в ходе его выполнения.
- *Participant* – участник процесса, производящий «действия» над объектами и осуществляющий переходы (участники могут являться как человеческими, так и машинными ресурсами).
- *Application* – внешнее IT- или другое приложение, используемое для выполнения «действий».

Стандарт WfMC определяет три фундаментальных понятия:

- *Workflow* – это процесс, определяемый как автоматизация некоторого делового процесса. Он включает в себя логику процесса и правила маршрутизации.
- *Workflow Management* – это набор методов и технологий для автоматизации бизнес процесса в целом или его части посредством информационных объектов.
- *Workflow Management System (WfMS)* - это система, которая определяет, создает и управляет выполнением workflow-процессов с помощью специального программного обеспечения.

К характерным особенностям XPDL можно отнести следующее:

- С позиций описания XPDL рабочий процесс представляет собой направленный граф, узлами которого являются «действия», связанные между собой переходами. Переходы могут быть условными, причем условие проверяется на этапе выполнения конкретного «действия».
- В языке XPDL существует возможность выделения «блоков» – возможность объединения «действий» в блок «действий» со своими отдельными условными или безусловными точками входа и выхода. Также имеется возможность определять вложенные подпроцессы внутри родительского процесса, которые сами по себе представляют полноценные потоки работ.
- Спецификация поддерживает возможность экспорта некоторых блоков описания одного процесса в описание другого с возможностью переопределения части

импортируемого описания, что исключает необходимость дублирования идентичных фрагментов описания в нескольких процессах.

- XPDL является расширяемым стандартом. Он позволяет определять набор элементов и атрибутов, специфичных для конкретной сферы его применения.
- Элементы описания процессов XPDL имеют обширный набор атрибутов, определяющих ход выполнения процесса. К ним можно отнести условные выражения для переходов, временные рамки, задание множественных исполнителей «действий» и т.д.

4.7.2.2The Business Process Modeling Language (BPML) и XLANG

Спецификации BPML и XLANG, предложенные Business Process Management Initiative и корпорацией Microsoft соответственно, являются родственными стандартами, реализующими блочную модель потоков работ. Рассмотрим спецификации данного семейства на примере BPML ввиду того, что данный стандарт охватывает наиболее широкий спектр характерных особенностей моделей рабочих процессов данного типа.

Business Process Modeling Language представляет собой абстрактную модель и XML-язык для описания рабочих процессов различного типа. В качестве исполнителей заданий потока работ в модели BPML выступают различного рода Web-сервисы. Для описания сервиса и всех услуг, предоставляемых им потребителю, используется спецификация WSDL (Web Services Description Language). Общение с такими службами может осуществляться, например, посредством протокола SOAP (Simple Object Access Protocol).

В отличие от спецификации XPDL, стандарт BPML представляет собой блочную структуру. Элементарными блоками, из которых строится модель рабочего процесса в BPML, являются *«действия»* (*Activity*). С позиций подобного блочного подхода сам рабочий процесс (в отличие от XPDL) не выделяется в виде отдельного объекта, а является просто специальным случаем агрегированного действия. В спецификации BPML выделяется три специальных типа процессов:

- Вложенный процесс (Nested Process). Данный тип процесса, как и любое «действие», может быть включен в иерархию действий родительского (или на верхнем уровне корневого) процесса. При этом он образует свой собственный контекст.
- Процесс, обрабатывающий исключительные события (Exception Process). Процесс этого типа, будучи указан в контексте родительского процесса, выполняет обработку системных событий, объявленных как исключительные.

- Процесс-компенсатор (Compensation Process). После завершения работы любого процесса (с ошибкой или нет) процесс данного типа позволяет проводить действия по освобождению системных ресурсов и другие завершающие шаги.

Для общения между процессами (или «действиями») в спецификации BPML используются *сообщения* (*Messages*) и *сигналы* (*Signals*). Для этого процессы (или агрегированные «действия») должны назначать обработчиков сообщений или сигналов, а «действие»-источник должно генерировать сигналы (сообщения).

Двумя важными компонентами модели BPML являются *системные ошибки* (*Faults*) и *расписания* (*Schedules*). Системные ошибки (как и исключительные события) приводят к завершению выполнения текущего задания, в результате чего управление передается специальным обработчикам (системным или определенным в контексте). Расписания выполняют управляющие функции: в их задачу входит генерация специальных сообщений в определенные моменты времени, что позволяет осуществлять планирование запуска процессов.

Аналогом Workflow Relevant Data в XPDL (хранилище оперативных данных) в случае BPML является иерархия вложенных контекстов рабочих процессов.

4.7.2.3 The Web Services Flow Language (WSFL)

The Web Services Flow Language, разработанный компанией IBM, представляет собой XML-язык, описывающий композицию произвольного типа Web-служб в рамках одной модели потоков (Flow Model). Данная композиция описывается последовательностью точек доступа к функциям, предоставляемым различными службами. Порядок запуска сервисов определяется с помощью управляющих потоков и потоков данных между Web-службами.

Каждый поток работ, объединяющий в себе использование набора некоторых Web-служб, включает в себя:

- Бизнес процесс, определяющий набор действий, которые необходимо выполнить в течение данного процесса.
- Управляющие правила, которые определяют последовательность выполнения шагов бизнес-процесса.
- Поток информации между различными этапами бизнес-процесса.

Формальная WSFL-модель, отражающая взаимодействие между потоком работ и внешними поставщиками услуг, представлена на следующей диаграмме:

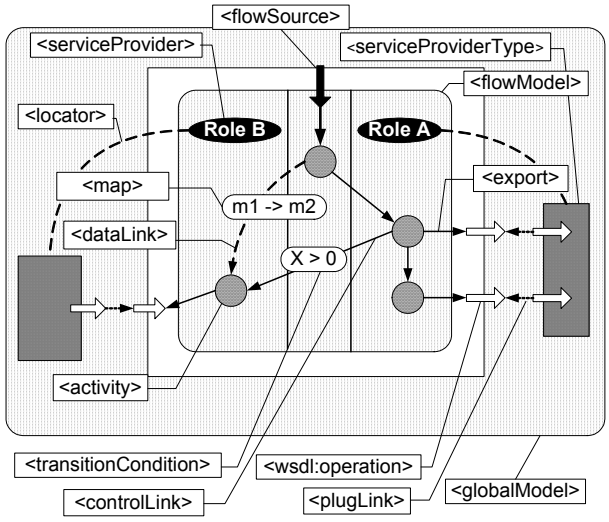


Рисунок 14. Структура WSFL-композиции WEB-сервисов

WSFL Flow Model определяет структуру бизнес-процесса: *«действия»* (помечены кружками на диаграмме) описывают шаги процесса, а *«управляющие связи»* (элемент <**controlLink>**) и *«потоки данных»* (элемент <**dataLink>**) определяют последовательность переходов и потоки данных между «действиями» соответственно. Для каждого «действия» назначен *«поставщик услуг»* (элемент <**serviceProvider>**), ответственный за выполнение данного этапа бизнес-процесса, и определяются связи между «действиями» в модели потоков и *«операциями»* (элемент **<wsdl:operation>**), предоставляемыми поставщиками услуг (элементы **<export>** и **<plugLink>)**. Результирующая модель потоков работ (элемент **<flowModel>**) изображена в центре диаграммы. Она разбита вертикальными линиями на части, отражая тем самым связь «действий» процесса с конкретными поставщиками услуг.

Приведем список характерных особенностей спецификации WSFL:

- Этапы бизнес-процесса представлены «действиями», которые связываются в направленный граф с условными или безусловными переходами между его вершинами. Каждое «действие» связано с некоторой операцией провайдера услуг, отвечающего за его выполнение.
- Связи между «действиями» (Control Link) представляют собой направленные дуги графа маршрута модели потоков. Переходы между «действиями» могут быть условными.
- Модель WSFL допускает возможность параллельного выполнения нескольких «действий». Для этого служит «действие» специального типа, называемое *fork activity (разветвление)*, переход с которого может осуществляться одновременно

на несколько других действий. При этом WSFL поддерживает набор средств для синхронизации одновременно выполняемых этапов процесса.

- Существует два выделенных типа «действий»: начальные (на которые не осуществляются переходы посредством использования Control Link) и конечные (которые не связаны «исходящими» переходами с другими «действиями» процесса).
- Между последовательно выполняющимися «действиями» может быть организован поток данных для обмена управляющей и другой информацией.

Языки WSFL и XPDL относятся к одному семейству графовых XML-языков описания потоков работ, в отличие от блочной структуры BPML, что определяет их схожую функциональность, хотя XPDL и дает больший уровень абстракции.

4.7.2.4Business Process Execution Language For Web-Services (BPEL4WS)

Язык описания рабочих процессов Business Process Execution Language for Web Services (BPEL4WS) представляет собой объединение спецификаций WSFL и XLANG, и поэтому стал основным стандартом описания композиций Web-сервисов. BPEL4WS объединяет в себе лучшие стороны WSFL (поддержка графовых потоков работ) и XLANG (блочная структура процесса) позволяя описать любой бизнес-процесс в самой естественной форме. Помимо того, что BPEL4WS является языком описания реализаций бизнес-процессов, он так же может использоваться для описания интерфейсов бизнес-процессов, используя понятие «абстрактного процесса».

4.7.2.4.1 Базовая концепция BPEL4WS

BPEL4WS поддерживает два различных сценария использования:

- Реализация исполняемых бизнес-процессов.
- Описание неисполняемых «абстрактных» процессов.

Роль BPEL4WS как языка описания реализаций исполняемых бизнес-процессов это определение нового Web-сервиса путем композиции набора существующих сервисов. Таким образом, BPEL4WS в своей основе является языком описания таких композиций. Интерфейс композиции Web-сервисов определяется как коллекция типов портов WSDL (portTypes) – точно так же как и любой другой Web-сервис. Композиция (называемая *процесс*) определяет, как интерфейсы Web-сервисов вписываются в процесс запуска всей композиции.

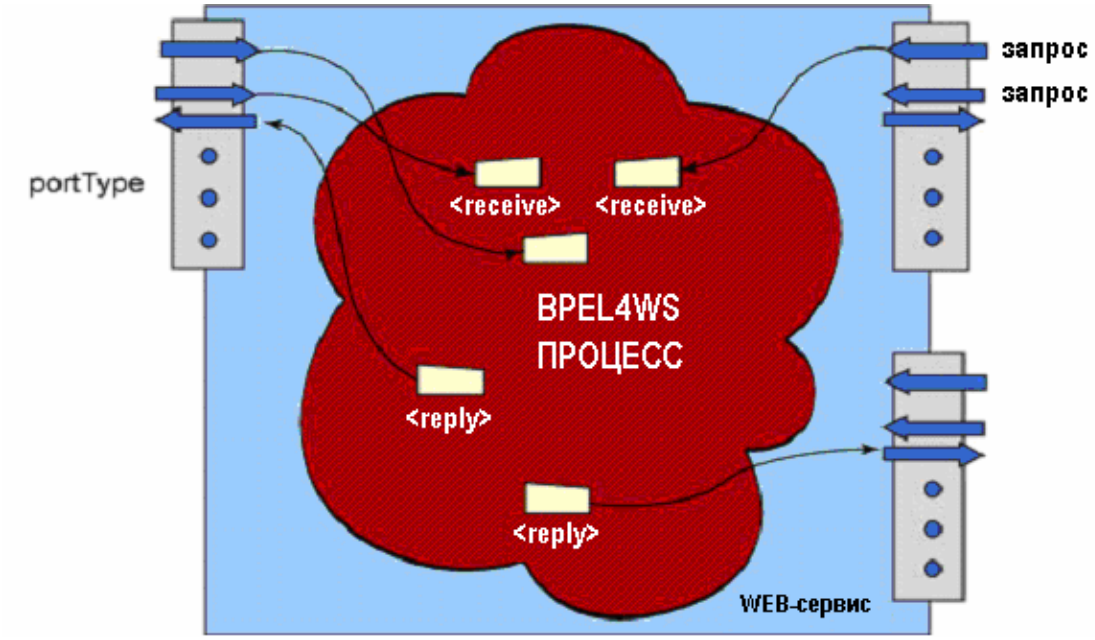


Рисунок 15. Внешний интерфейс BPEL4WS-процесса

4.7.2.4.2 Реализация BPEL4WS в виде WEB-сервиса

В отличие от традиционного языка программирования при реализации WSDL-интерфейса на языке BPEL4WS каждая операция каждого типа порта не отображается на отдельный логический модуль на языке BPEL4WS. Вместо этого процесс на этом языке описывает весь сервис в целом. Таким образом, специфичные «точки входа», соответствующие внешним вызовам операций интерфейса пользователями, определяются в пределах BPEL4WS описания. Эти точки входа либо соответствую WSDL операциям «только-запрос», либо операциям «запрос-ответ». В последнем случае процесс должен так же определять, где и как генерируются ответные сообщения. BPEL4WS использует и поддерживает только операции типа «только-запрос» или запрос-ответ»; операции типа «только-ответ» и «ответ-запрос» не поддерживаются и не требуются для реализации бизнес-процессов на языке BPEL4WS.

BPEL4WS процесс обычно представляется поточной диаграммой как способ представления алгоритма. Каждый шаг процесса называется *действием*. Существует коллекция примитивных действий: вызов операции некоторого Web-сервиса (*<invoke>*), ожидание внешнего вызова операции WSDL-интерфейса процесса (*<receive>*), генерация ответа на запрос (*<reply>*), ожидание в течение некоторого периода времени (*<wait>*), копирование данных из одной переменной в другую (*<assign>*), сигнализация об ошибках (*<throw>*), остановка всего процесса (*<terminate>*), или пустое действие (*<empty>*).

Эти примитивные действия могут комбинироваться в более сложные алгоритмы, используя структурированные действия, описываемые языком. К таким действиям относится возможность определять упорядоченную последовательность действий (*<sequence>*), возможность определения условных выборов (*<switch>*), возможность определения циклов (*<while>*), возможность направить процесс по одному из альтернативных путей (*<pick>*), и, наконец, возможность указать, что набор действий должен выполняться параллельно (*<flow>*). При параллельном исполнении действий можно определять порядок их активации посредством использования элементов <link>.

BPEL4WS позволяет рекурсивно комбинировать структурированные действия для того, чтобы выразить произвольно сложный алгоритм, представляющий реализацию Web-сервиса.

4.7.2.4.3 Взаимодействие с участниками процесса

Как язык описания композиций набора сервисов в новый сервис, BPEL4WS определяет набор вызовов других сервисов и/или получение обратных вызовов от них. Первая из упомянутых задач выполняется посредством использования *<invoke>*, а последнее – посредством использования действий *<receive>* и *<reply>*. BPEL4WS производит вызовы сторонних сервисов, которые представляют собой *партнеров* процесса. Таким образом, партнеры представляют собой сервисы, к которым процесс обращается в рамках общего алгоритма, или те, которые сами вызывают те или иные операции процесса.

4.7.2.4.4 Обработка исключений

Разработчикам необходимы средства обработки исключительных ситуаций на маршруте бизнес-процесса. BPEL4WS имеет средства генерации и обработки исключений, предоставляемые конструкциями *<throw>* и *<catch>*. Понятие исключений в BPEL4WS напрямую связано с понятием ошибок в WSDL и, фактически, основано на нем.

В добавок, BPEL4WS поддерживает понятие *компенсации*, которое является механизмом, позволяющим разработчику процесса реализовать компенсирующие действия для конкретных «необратимых» действий. Например, можно представить себе процесс резервирования путевки. Если резервирование не состоялось, необходимо предпринять действия по его отмене. Эти действия называются «компенсирующими» по отношению к изначально совершенным действиям.

Обработка исключений и компенсация поддерживаются рекурсивно на уровне BPEL4WS посредством введения понятия контекстов, которые представляют собой блок процесса, определяющий своих обработчиков ошибок и компенсирующие действия.

4.7.2.4.5 Жизненный цикл процесса

Web-сервис реализуемый процессом на языке BPEL4WS имеет «экземплярную» модель жизненного цикла. Это означает, что клиент таких сервисов всегда взаимодействует с конкретным экземпляром сервиса (процесса).

В отличие от традиционных систем распределенных объектов, в BPEL4WS новые экземпляры процесса создаются в ответ на получение сообщения, адресованного сервису. Это означает, что экземпляры идентифицируются не только с помощью идентификатора процесса, но и некоторыми ключевыми полями в пределах поступившего сообщения. Например, если процесс представляет собой систему оформления заказа, номер накладной мог бы являться ключевым полем для идентификации конкретного экземпляра взаимодействий. Таким образом, если соответствующий экземпляр процесса недоступен при получении сообщения в «стартовой» точке процесса, новый экземпляр процесса автоматически создается и ассоциируется с ключевыми данными сообщения. Сообщения могут приниматься процессом в «не стартовых» точках, только если был обнаружен подходящий экземпляр процесса; это означает, что все сообщения всегда доставляются конкретным экземплярам процесса. В BPEL4WS процесс определения подходящего экземпляра или создание нового в случае необходимости называется *корреляцией сообщений*.

4.7.2.5The Web Services Choreography Interface (WSCI)

Спецификация The Web Services Choreography Interface была предложена Sun, SAP, BEA и Intalio. Она определяет XML-язык описания взаимодействий Web-сервисов. При этом язык описывает весь процесс хореографии, включая сообщения, передаваемые между участниками взаимодействия. Спецификация поддерживает корреляцию сообщений, правила следования, обработку исключений, транзакции и динамическое взаимодействие.

Ключевым аспектом спецификации WSCI является то, что она описывает только видимое поведение Web-сервисов. WSCI не описывает исполняемые бизнес-процессы, как, например, BPEL4WS. Более того, отдельный документ WSCJ описывает участие только одного из взаимодействующих Web-сервисов в процессе обмена сообщениями.

Следующая диаграмма иллюстрирует хореографию WSCI , которая включает множество документов – по одному на каждого участника процесса. В WSCI не существует единого контролирующего процесса, управляющего взаимодействием.

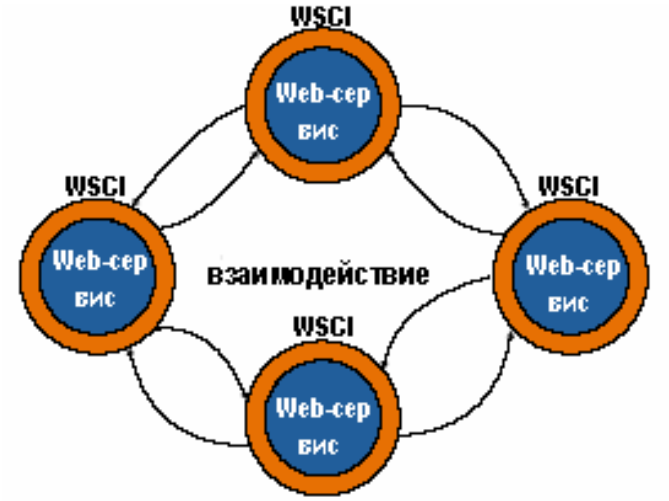


Рисунок 16. WSCI-хореография

WSCI можно так же рассматривать как пласт над стеком существующих Web-сервисов. Каждое действие в WSCI определяет единицу работы, которая обычно отображается на конкретную WSDL-операцию. WSCI можно представить как связующий элемент WSDL, описывающий, как может строиться хореография операций. Другими словами, WSDL используется для описания точек входа для всех доступных сервисов, а WSDL описывает взаимодействие между этими WSDL-операциями, аналогично тому, как BPEL4WS использует WSDL.

WSCI поддерживает как базовые, так и структурированные действия:

- Конструкция <action> для описания базового запроса или ответа на вызов. Каждое действие определяет связанную WSDL-операцию у роль участника взаимодействия. Внешние сервисы могут быть вызваны с помощью конструкции <call>.
- WSCI поддерживает большое количество структурированных действий, включая последовательное и параллельное исполнение и условный цикл. WSCI так же определяет действие <all>, которое указывает, что необходимо выполнить некоторый набор действий, но без указания какого-либо порядка выполнения.

Следующий листинг иллюстрирует простой пример WSCI. Процесс закупки содержит два последовательных действия: «получение заказа» и «подтверждение». Каждое

действие отображается на тип порта WSDL. Между двумя шагами процесса устанавливается корреляция. Данный документ WSCI описывает поведение посредника. Таким образом, в процессе должны быть еще определены и документы с описанием поведения заказчика и поставщика.

```
<process name=”Purchase” instantiation=”message”>
  <sequence>
        <action name=”ReceiveOrder” role=”Agent” operation=”tns:Order”>
        </action>
        <action name=”Confirm” role=”Agent” operation=”tns:Confirm”>
              <correlate correlation=”tns:ordered”/>
              <call process=”tns:Purchase”/>
        </action>
  </sequence>
</process>
```

WSCI поддерживает понятие бизнес-транзакций и обработки исключений. В рамках WSCI может быть определен контекст транзакции, аналогичный действию <scope> BPEL4WS-процесса. Если внутри контекста определен ряд действий, любая ошибка приведет к тому, что результаты выполнения всей группы действий будут возвращены в исходное состояние (“откат”).

4.7.3Выводы

Язык описания АРП в распределенной среде Web-сервисов BPEL4WS обладает рядом архитектурных преимуществ, к основным из которых можно отнести:

- отражение специфики как графовой, так и блочной структуры описания рабочих процессов;
- возможность представления всего рабочего процесса в виде Web-сервиса для определения композиций процессов;
- абстрактность и динамичность связи с реальными исполнителями того или иного этапа процесса (конкретными Web-сервисами);
- возможность привлечения человеческих ресурсов во время исполнения процесса за счет поддержки асинхронной модели взаимодействия;
- наличие развитых механизмов управления экземплярами процессов и маршрутизации внешних сообщений между ними;

- развитые механизмы обработки исключительных ситуаций и компенсации контекстов, позволяющие реализовать средствами самого языка модель бизнес-транзакций.

В настоящий момент данная спецификация проходит стандартизацию в рамках международного консорциума OASIS. При этом средства поддержки BPEL4WS-процессов уже реализованы рядом крупнейших производителей программного обеспечения, такими, как Microsoft (BizTalk 2004) и Oracle (BPEL Process Manager), что позволяет осуществлять интеграцию различных систем, использующих различные платформы и программное обеспечение, в рамках единой модели описания бизнес-процессов.

Кроме того, стандарт рекомендован к применению совместно с другими последними стандартами в области рабочих процессов, такими как WS-Transaction (протоколы координации транзакций Web-сервисов), WS-Security (безопасный протокол передачи данных в рамках процесса, гарантирующий их целостность) и т.д. Workflow-процесс, описанный на языке BPEL4WS, позволяет реализовать всю логику обработки пользовательских запросов и управления взаимодействием с распределенными системами.

Таким образом, язык описания АРП BPEL4WS включает все необходимые средства для создания транзакционных, безопасных и сложных сценариев электронного взаимодействия Web-сервисов.

4.8 Глобальная идентификация

4.8.1.1Постановка задачи

Задача ставится аналогично задачи глобальной идентификации в общем контексте (2.8.1)

4.8.1.2Обзор и анализ

Все рассмотренные в разделе 2.8.2 стандарты применимы для работы совместно с Web-технологиями.

4.8.1.3Выводы

В соответствии с разделом 2.8.3 выбор следует остановить на Handle System.

5 Общие рекомендации по использованию стандартов в рамках СУВ

В этом разделе приводятся общие рекомендации по использованию интеграционных механизмов WSA. При этом описываются существующие неоднозначности спецификаций и предлагаются шаблоны оформления соответствующих элементов, обеспечивающие эквивалентное понимание этих сложных мест спецификаций различными ИС. Так же приводятся некоторые аспекты практического использования стандартов в рамках СУВ: инкапсуляция сообщений в транспортные протоколы, необходимые расширения спецификаций, соглашения по стилю оформления документов и т.д.

Помимо общих рекомендации к использованию стандартов в данном разделе приводятся выдержки из профиля стандартов международного стандарта интероперабельности web-сервисов WS-I BasicProfile-1.0. Данный стандарт призван разрешить неоднозначности толкования отдельных элементов спецификаций WSA для повышения интероперабельности. Помимо этого он определяет совместимость версий стандартов WSA. Данный профиль стандартов одобрен ведущими разработчиками платформ поддержки Web-сервисов и постоянно развивается и модифицируется. Ввиду этого большая часть рекомендаций этого раздела приведена в соответствии с форматом профиля стандартов WS-I BasicProfile-1.0 в виде рекомендаций по использованию отдельных конструкций и примеров правильного и неправильного оформления соответствующих документов, а не в виде обычного изложения, характерного для большей части данного документа. В связи с этим конструкции вида:

R2028 ОПИСАНИЕ, использующее пространство имен WSDL (с префиксом "wsdl" в BasicProfile-1.0) ДОЛЖНО быть правильным в соответствии со спецификациями XML-схем, приведенными в "http://schemas.xmlsoap.org/wsdl/2003-02-11.xsd".

представляют собой непосредственные рекомендации по использованию тех или иных элементов спецификации. Для поддержки контекста документа WS-I BasicProfile-1.0 у подобных рекомендаций сохранена оригинальная нумерация вида: *R2028* (рекомендация №2028 в документе WS-I BasicProfile-1.0.

5.1 ИС и сервисы

5.1.1 Рекомендации по оформлению WSDL-описаний Web-сервисов

BasicProfile-1.0 использует язык WSDL для описания Web-сервисов как наборов «конечных адресных точек», работающих с сообщениями.

Раздел BasicProfile-1.0, описывающий рекомендации по использованию WSDL, включил в себя с некоторыми расширениями следующие спецификации:

- Web Services Description Language (WSDL) 1.1

  Расширения:
  - WSDL-расширения – WSDL в некоторых случаях расширяющие элементы; использование таких расширений требует специальных соглашений, выходящих за рамки стандарта.
  - Относительные URI – WSDL неадекватно специфицирует использование относительных URI, их применение требует дальнейших согласований (см. дополнительную информацию в XML-Base).
  - Виды проверок – выполняет ли анализатор, считывающий WSDL- и XML-Schema-документы, DTD-проверки.
  - Вызов внешних ресурсов - выполняет ли анализатор, считывающий WSDL- и XML-Schema-документы, вызов внешних объектов и DTD.

- XML Schema Part 1: Structures

  Расширения:
  - Примечания в схеме – XML-схема допускает примечания, которые могут использоваться для сообщения дополнительной информации о структурах данных.

- XML Schema Part 2: Datatypes

5.1.1.1 Структура документа WSDL

В разделе BasicProfile-1.0, посвященном рекомендациям по оформлению структуры документа WSDL, используются следующие спецификации (некоторые их разделы):

- WSDL 1.1, Section 2.1

WSDL 1.1 определяет основанную на XML структуру для описания Web-сервисов. BasicProfile-1.0 утверждает обязательность этой структуры и выдвигает следующие ограничения на ее использование:

5.1.1.1.1 Определения WSDL-схем

Нормативные схемы WSDL, приведенные в Приложении 4 спецификаций WSDL 1.1, содержат расхождения с нормативным текстом спецификаций. BasicProfile-1.0 ссылается на новые схемные документы, в которых исправлены уже известные ошибки.

R2028 ОПИСАНИЕ, использующее пространство имен WSDL (с префиксом "wsdl" в BasicProfile-1.0) ДОЛЖНО быть правильным в соответствии со спецификациями XML-схем, приведенными в "http://schemas.xmlsoap.org/wsdl/2003-02-11.xsd".

R2029 ОПИСАНИЕ, использующее пространство имен WSDL связи с SOAP (с префиксом " soapbind " в BasicProfile-1.0) ДОЛЖНО быть правильным в соответствии со спецификациями XML-схем, приведенными в " http://schemas.xmlsoap.org/wsdl/soap/2003-02-11.xsd ".

Хотя BasicProfile-1.0 и требует правильности WSDL-описаний в соответствии со схемами, он не требует, чтобы пользователи проверяли правильность WSDL-документов. Ответственность за правильность WSDL-документов возлагается на их авторов.

5.1.1.1.2 WSDL и импорт схем

Некоторые примеры в WSDL 1.1 некорректно представляют WSDL-конструкцию импорта, используемую для импорта определений XML-схем. BasicProfile-1.0 уточняет использование механизмов импорта для обеспечения их непротиворечивости и ограниченности соответствующими областями определения. Импортируемые документы XML-схем ограничены также версией XML и требованиями кодирования, соответствующим тем, которые используются в импортируемых WSDL-документах.

R2001 ОПИСАНИЕ ДОЛЖНО использовать только WSDL-утверждение "import" для импорта другого WSDL-описания.

R2002 Для импорта определений XML-схемы ОПИСАНИЕ ДОЛЖНО использовать XML -утверждение "import".

R2003 ОПИСАНИЕ ДОЛЖНО использовать утверждение "import" XML-схемы только в элементе xsd:schema секции типов.

R2004 ОПИСАНИЕ НЕ ДОЛЖНО использовать утверждение "import" XML-схемы для импорта какой-либо схемы из документа, корневым элементом которого не является "schema" из пространства имен "http://www.w3.org/2001/XMLSchema".

R2009 XML-схема, прямо или косвенно импортируемая ОПИСАНИЕМ, МОЖЕТ включать Unicode-маркер порядка байтов (Unicode Byte Order Mark-BOM).

R2010 XML-схема, прямо или косвенно импортируемая ОПИСАНИЕМ, ДОЛЖНА использовать кодировку UTF-8 или UTF-16.

R2011 XML-схема, прямо или косвенно импортируемая ОПИСАНИЕМ, ДОЛЖНА использовать версию 1.0 рекомендаций XML W3C.

Например:

НЕПРАВИЛЬНО:

```
<definitions name="StockQuote"
   targetNamespace="http://example.com/stockquote/definitions"
   xmlns:xsd1="http://example.com/stockquote/schemas""
                 ...
   xmlns="http://schemas.xmlsoap.org/wsdl/">

   <import namespace="http://example.com/stockquote/schemas"
         location="http://example.com/stockquote/stockquote.xsd"/>

   <message name="GetLastTradePriceInput">
        <part name="body" element="xsd1:TradePriceRequest"/>
    </message>
               ...
</definitions>
```

ПРАВИЛЬНО:

```
<definitions name="StockQuote"
      targetNamespace="http://example.com/stockquote/definitions">
      <import namespace="http://example.com/stockquote/definitions"
           location="http://example.com/stockquote/stockquote.wsdl"/>
      <message name="GetLastTradePriceInput">
         <part name="body" element="..."/>
      </message>
                  ...
   </definitions>
```

ПРАВИЛЬНО:

```
<definitions name="StockQuote"
   targetNamespace="http://example.com/stockquote/"
```

```
   xmlns:xsd1="http://example.com/stockquote/schemas"
                  ...
   xmlns="http://schemas.xmlsoap.org/wsdl/">

   <import namespace="http://example.com/stockquote/definitions"
        location="http://example.com/stockquote/stockquote.wsdl"/>

   <message name="GetLastTradePriceInput">
      <part name="body" element="xsd1:TradePriceRequest"/>
   </message>
               ...
</definitions>
```

5.1.1.1.3 Синтаксис WSDL-атрибута указателя импорта WSDL и импорт схем

Некоторые примеры в WSDL 1.1 некорректно представляют WSDL-конструкцию импорта, используемую для импорта определений XML-схем. BasicProfile-1.0 уточняет использование механизмов импорта для обеспечения их непротиворечивости и ограниченности соответствующими областями определения. Импортируемые документы XML-схем ограничены также версией XML и требованиями кодирования, соответствующим тем, которые используются в импортируемых WSDL-документах.

R2001 ОПИСАНИЕ ДОЛЖНО использовать только WSDL-утверждение "import" для импорта другого WSDL-описания.

R2002 Для импорта определений XML-схемы ОПИСАНИЕ ДОЛЖНО использовать XML -утверждение "import".

R2003 ОПИСАНИЕ ДОЛЖНО использовать утверждение "import" XML-схемы только в элементе xsd:schema секции типов.

R2004 ОПИСАНИЕ НЕ ДОЛЖНО использовать утверждение "import" XML-схемы для импорта какой-либо схемы из документа, корневым элементом которого не является "schema" из пространства имен "http://www.w3.org/2001/XMLSchema".

R2009 XML-схема, прямо или косвенно импортируемая ОПИСАНИЕМ, МОЖЕТ включать Unicode-маркер порядка байтов (Unicode Byte Order Mark-BOM).

R2010 XML-схема, прямо или косвенно импортируемая ОПИСАНИЕМ, ДОЛЖНА использовать кодировку UTF-8 или UTF-16.

R2011 XML-схема, прямо или косвенно импортируемая ОПИСАНИЕМ, ДОЛЖНА использовать версию 1.0 рекомендаций XML W3C.

Например:

НЕПРАВИЛЬНО:

```
<definitions name="StockQuote"
   targetNamespace="http://example.com/stockquote/definitions"
   xmlns:xsd1="http://example.com/stockquote/schemas""
                  ...
   xmlns="http://schemas.xmlsoap.org/wsdl/">

   <import namespace="http://example.com/stockquote/schemas"
         location="http://example.com/stockquote/stockquote.xsd"/>

   <message name="GetLastTradePriceInput">
        <part name="body" element="xsd1:TradePriceRequest"/>
    </message>
                ...
</definitions>
```

ПРАВИЛЬНО:

```
<definitions name="StockQuote"
      targetNamespace="http://example.com/stockquote/definitions">
      <import namespace="http://example.com/stockquote/definitions"
           location="http://example.com/stockquote/stockquote.wsdl"/>
      <message name="GetLastTradePriceInput">
         <part name="body" element="..."/>
      </message>
                  ...
   </definitions>
```

ПРАВИЛЬНО:

```
<definitions name="StockQuote"
   targetNamespace="http://example.com/stockquote/"
   xmlns:xsd1="http://example.com/stockquote/schemas"
                  ...
   xmlns="http://schemas.xmlsoap.org/wsdl/">

   <import namespace="http://example.com/stockquote/definitions"
        location="http://example.com/stockquote/stockquote.wsdl"/>
```

```
   <message name="GetLastTradePriceInput">
      <part name="body" element="xsd1:TradePriceRequest"/>
   </message>
                ...
</definitions>
```

В WSDL 1.1 недостаточно четко определено, является ли атрибут location элемента wsdl:import обязательным и каково должно быть его содержание.

R2007 ОПИСАНИЕ ДОЛЖНО содержать непустой атрибут location в элементе wsdl:import.

Хотя представление элемента wsdl:import составлено по образцу `xsd:import`, атрибут location является обязательным в wsdl:import, в то время как соответствующий атрибут `schemaLocation` в xsd:import является необязательным. В соответствии с принятой обязательностью атрибута location, его содержание не должно быть пустым.

5.1.1.1.4 Семантика WSDL-атрибута указателя импорта WSDL и импорт схем

Некоторые примеры в WSDL 1.1 некорректно представляют WSDL-конструкцию импорта, используемую для импорта определений XML-схем. BasicProfile-1.0 уточняет использование механизмов импорта для обеспечения их непротиворечивости и ограниченности соответствующими областями определения. Импортируемые документы XML-схем ограничены также версией XML и требованиями кодирования, соответствующим тем, которые используются в импортируемых WSDL-документах.

R2001 ОПИСАНИЕ ДОЛЖНО использовать только WSDL-утверждение "import" для импорта другого WSDL-описания.

R2002 Для импорта определений XML-схемы ОПИСАНИЕ ДОЛЖНО использовать XML -утверждение "import".

R2003 ОПИСАНИЕ ДОЛЖНО использовать утверждение "import" XML-схемы только в элементе xsd:schema секции типов.

R2004 ОПИСАНИЕ НЕ ДОЛЖНО использовать утверждение "import" XML-схемы для импорта какой-либо схемы из документа, корневым элементом которого не является "schema" из пространства имен "http://www.w3.org/2001/XMLSchema".

R2009 *XML-схема, прямо или косвенно импортируемая ОПИСАНИЕМ, МОЖЕТ включать Unicode-маркер порядка байтов (Unicode Byte Order Mark-BOM).*

R2010 *XML-схема, прямо или косвенно импортируемая ОПИСАНИЕМ, ДОЛЖНА использовать кодировку UTF-8 или UTF-16.*

R2011 *XML-схема, прямо или косвенно импортируемая ОПИСАНИЕМ, ДОЛЖНА использовать версию 1.0 рекомендаций XML W3C.*

Например:

НЕПРАВИЛЬНО:

```
<definitions name="StockQuote"
   targetNamespace="http://example.com/stockquote/definitions"
   xmlns:xsd1="http://example.com/stockquote/schemas""
                 ...
   xmlns="http://schemas.xmlsoap.org/wsdl/">

   <import namespace="http://example.com/stockquote/schemas"
         location="http://example.com/stockquote/stockquote.xsd"/>

   <message name="GetLastTradePriceInput">
        <part name="body" element="xsd1:TradePriceRequest"/>
    </message>
               ...
</definitions>
```

ПРАВИЛЬНО:

```
<definitions name="StockQuote"
      targetNamespace="http://example.com/stockquote/definitions">
      <import namespace="http://example.com/stockquote/definitions"
           location="http://example.com/stockquote/stockquote.wsdl"/>
      <message name="GetLastTradePriceInput">
         <part name="body" element="..."/>
      </message>
                  ...
</definitions>
```

ПРАВИЛЬНО:

```
<definitions name="StockQuote"
   targetNamespace="http://example.com/stockquote/"
   xmlns:xsd1="http://example.com/stockquote/schemas"
                 ...
   xmlns="http://schemas.xmlsoap.org/wsdl/">
```

```
   <import namespace="http://example.com/stockquote/definitions"
        location="http://example.com/stockquote/stockquote.wsdl"/>

   <message name="GetLastTradePriceInput">
      <part name="body" element="xsd1:TradePriceRequest"/>
   </message>
                ...
</definitions>
```

5.1.1.1.5 Расположение элементов wsdl:import

Пример 3 в секции 3.1 WSDL вносит неясность в вопрос расположения элементов wsdl:import.

R2022 Элементоыв wsdl:import в ОПИСАНИИ должны предшествовать всем другим элементам из пространства имен WSDL, кроме wsdl:documentation.

R2023 Элементов wsdl:types в ОПИСАНИИ должны предшествовать всем другим элементам из пространства имен WSDL, кроме wsdl:documentation и wsdl:import.

Например,

НЕПРАВИЛЬНО:

```
<definitions name="StockQuote"
                  ...
   xmlns="http://schemas.xmlsoap.org/wsdl/">

   <import namespace="http://example.com/stockquote/definitions"
         location="http://example.com/stockquote/stockquote.wsdl"/>

   <message name="GetLastTradePriceInput">
       <part name="body" type="tns:TradePriceRequest"/>
   </message>
                ...
   <service name="StockQuoteService">
      <port name="StockQuotePort" binding="tns:StockQuoteSoap">
           ....
      </port>
   </service>
```

```
   <types>
      <schema targetNamespace="http://example.com/stockquote/schemas"
               xmlns="http://www.w3.org/2001/XMLSchema">
           .......
      </schema>
   </types>
</definitions>
```

ПРАВИЛЬНО:

```
<definitions name="StockQuote"
    targetNamespace="http://example.com/stockquote/definitions">
     <import namespace="http://example.com/stockquote/base"
       location="http://example.com/stockquote/stockquote.wsdl"/>

      <message name="GetLastTradePriceInput">
         <part name="body" element="..."/>
      </message>
                  ...
</definitions>
```

ПРАВИЛЬНО:

```
<definitions name="StockQuote"
                  ...
   xmlns="http://schemas.xmlsoap.org/wsdl/">

  <types>
     <schema targetNamespace="http://example.com/stockquote/schemas"
          xmlns="http://www.w3.org/2001/XMLSchema">
           .......
     </schema>
   </types>

   <message name="GetLastTradePriceInput">
        <part name="body" element="tns:TradePriceRequest"/>
   </message>
              ...
   <service name="StockQuoteService">
      <port name="StockQuotePort" binding="tns:StockQuoteSoap">
           ....
      </port>
   </service>
```

```
</definitions>
```

5.1.1.1.6 Требования к версии XML Расположение элементов wsdl:import

Пример 3 в секции 3.1 WSDL вносит неясность в вопрос расположения элементов wsdl:import.

R2022 Элементов wsdl:import в ОПИСАНИИ должны предшествовать всем другим элементам из пространства имен WSDL, кроме wsdl:documentation.

R2023 Элементов wsdl:types в ОПИСАНИИ должны предшествовать всем другим элементам из пространства имен WSDL, кроме wsdl:documentation и wsdl:import.

Например,

НЕПРАВИЛЬНО:

```
<definitions name="StockQuote"
                  ...
   xmlns="http://schemas.xmlsoap.org/wsdl/">

   <import namespace="http://example.com/stockquote/definitions"
         location="http://example.com/stockquote/stockquote.wsdl"/>

   <message name="GetLastTradePriceInput">
       <part name="body" type="tns:TradePriceRequest"/>
   </message>
              ...
   <service name="StockQuoteService">
      <port name="StockQuotePort" binding="tns:StockQuoteSoap">
           ....
      </port>
   </service>

   <types>
      <schema targetNamespace="http://example.com/stockquote/schemas"
               xmlns="http://www.w3.org/2001/XMLSchema">
           .......
      </schema>
   </types>
</definitions>
ПРАВИЛЬНО:
```

```
   <definitions name="StockQuote"
      targetNamespace="http://example.com/stockquote/definitions">

     <import namespace="http://example.com/stockquote/base"
       location="http://example.com/stockquote/stockquote.wsdl"/>

      <message name="GetLastTradePriceInput">
         <part name="body" element="..."/>
      </message>
                  ...
   </definitions>
```

ПРАВИЛЬНО:

```
<definitions name="StockQuote"
                  ...
   xmlns="http://schemas.xmlsoap.org/wsdl/">

  <types>
     <schema targetNamespace="http://example.com/stockquote/schemas"
          xmlns="http://www.w3.org/2001/XMLSchema">
           .......
     </schema>
   </types>

   <message name="GetLastTradePriceInput">
        <part name="body" element="tns:TradePriceRequest"/>
   </message>
              ...
   <service name="StockQuoteService">
      <port name="StockQuotePort" binding="tns:StockQuoteSoap">
           ....
      </port>
   </service>
</definitions>
```

Ни WSDL, ни XML Schema 1.0 не требуют определенной версии XML. Для обеспечения интероперабельности представляемые в XML WSDL-документы и импортируемые ими схемы должны использовать версию 1.0.

R4004 В ОПИСАНИИ должна использоваться версия 1.0 XML.

5.1.1.1.7 WSDL и Unicode BOM

XML 1.0 допускает, чтобы документы, использующие кодировку UTF-8, включали Unicode Byte Order Mark (BOM).

R4002 *ОПИСАНИЕ МОЖЕТ включать Unicode Byte Order Mark (BOM).*

5.1.1.1.8 Допустимые в WSDL кодировки

BasicProfile-1.0 требует, чтобы в SOAP и WSDL использовалась кодировка UTF-8 или UTF-16.

R4003 ОПИСАНИЕ ДОЛЖНО использовать кодировку UTF-8 или UTF-16.

5.1.1.1.9 Коэрция пространства имен

BasicProfile-1.0 запрещает коэрцию пространства имен в wsdl:import.

R2005 В импортируемом описании атрибут `targetNamespace` элемента `wsdl:definitions` должен иметь то же значение, что и атрибут `namespace` элемента `wsdl:import` в импортируемом ОПИСАНИИ.

5.1.1.1.10 WSDL documentation Element

Спецификации WSDL и WSDL Schema противоречивы в отношении расположения элементов `wsdl:documentation.`

R2020 *В ОПИСАНИИ элемент wsdl:documentation МОЖЕТ быть дочерним элемента wsdl:import.* *WSDL20*

R2021 *В ОПИСАНИИ элемент wsdl:documentation МОЖЕТ быть дочерним элемента wsdl: part.* *WSDL20*

R2024 *В ОПИСАНИИ элемент wsdl:documentation МОЖЕТ быть первым дочерним элемента wsdl: definitions.* *WSDL20*

5.1.1.1.11 Расширения WSDL

Поддержка расширений WSDL, не определенных явно в BasicProfile-1.0 или каком-либо другом WS-I-Профиле, может привести к проблемам интероперабельности в средствах разработки, в которых не была предусмотрена поддержка этих расширений.

R2025 ОПИСАНИЕ, содержащее WSDL-расширения, НЕ ДОЛЖНО использовать их, если они противоречат требованиям BasicProfile-1.0

R2026 ОПИСАНИЕ НЕ ДОЛЖНО содержать элементы расширения, со значением "true" атрибута wsdl:required (wsdl:binding, wsdl:portType, wsdl:message, wsdl:types или wsdl:import), WSDL-конструкций, которые должны соответствовать требованиям BasicProfile-1.0.

R2027 Если в процессе обработки некоторого элемента описания в пространстве имен WSDL процессор-клиент встречает среди дочерних элементов WSDL-расширение со значением "true" атрибута wsdl:required, которое процессор-клиент не распознает или не может обработать, процессор ДОЛЖЕН прекратить обработку этого элемента в пространстве имен WSDL.

В программных системах, которые считывают WSDL-описание и генерируют программное обеспечение Web-сервиса, могут отсутствовать средства распознавания неизвестных WSDL-расширений, поэтому следует избегать применения существенных WSDL-расширений. Использование некоторого существенного WSDL- расширения, для которого отсутствуют спецификации его использования и семантики, вызывает практически непреодолимые проблемы интероперабельности у всех, кроме самих авторов расширения. Использование некоторого существенного WSDL- расширения, для которого приведены спецификации его использования и семантики, снижает, но не снимает полностью связанные с ним проблемы интероперабельности.

Следующие элементы могут быть расширены только через атрибуты:

- `wsdl:import`
- `wsdl:part`
- `wsdl:portType`
- `wsdl:input` (в операции portType)
- `wsdl:output` (в операции portType)
- `wsdl:fault` (в операции portType)

Следующие элементы могут быть расширены через элементы или через атрибуты:

- `wsdl:definitions`
- `wsdl:types`
- `wsdl:message`
- `wsdl:operation`

- *wsdl:binding*
- *wsdl:input* (в операциях связывания)
- *wsdl:output* (в операциях связывания)
- *wsdl:fault* (в операциях связывания)
- *wsdl:service*
- *wsdl:port*

5.1.1.2Типы

В данном разделе BasicProfile-1.0 используются следующие спецификации (некоторые их разделы):

- WSDL 1.1, Section 2.2

Элемент *wsdl:types* спецификаций WSDL 1.1 содержит определения типов данных, существенных для описываемого Web-сервиса. BasicProfile-1.0 накладывает следующие ограничения на те части содержимого элементов *wsdl:types*, на которые ссылаются WSDL-элементы, требующие соответствия BasicProfile-1.0:

5.1.1.2.1 Qname-ссылки

XML Schema требует, чтобы каждая Qname-ссылка использовала или целевое пространство имен, или некоторое импортируемое пространство имен (явно указанное одним из элементов *xsd:import*). Qname-ссылки на пространство имен, представленные только вложенными импорт-указателями, не допускаются.

В WSDL 1.1 недостаточно четко указано, какие именно пространства имен целевой схемы относятся Qname-ссылки WSDL-элемента. BasicProfile-1.0 допускает Qname-ссылки из WSDL-элементов как на целевое пространство имен, определенное в элементе *xsd:schema*,так и на импортируемые пространства имен.

Как и в XML Schema, пространства имен, не указанные явно в WSDL-файле (через атрибут *targetNamespace* в *xsd:schema* или через атрибут пространства имен *namespace* *элемента* *xsd:import*) могут использоваться в Qname-ссылках. Qname-ссылки на пространство имен, представленные только вложенными импорт-указателями, не допускаются.

R2101 ОПИСАНИЕ НЕ ДОЛЖНО использовать Qname-ссылки на элементы пространства имен, которые не были ни импортированы, ни определены в содержащем их WSDL-документе.

R2102 Qname-ссылка на компонент схемы в ОПИСАНИИ ДОЛЖНА использовать пространство имен, указанное атрибутом `targetNamespace` *элемента* `xsd:schema`*, или пространство имен, указанное атрибутом* `namespace` *элемента* `xsd:import` *в элементе* `xsd:schema`*.*

5.1.1.2.2 Синтаксис целевого пространства имен схемы

Требование указания `targetNamespace` для всех элементов `xsd:schema` , которые являются потомками `wsdl:types`, является правильным стилем, представляет незначительные дополнительные усилия для авторов WSDL-документов и позволяет избежать не вполне определенных ситуаций.

R2105 Все элементы `xsd:schema`*, содержащиеся в элементе* `wsdl:types` *в ОПИСАНИИ ДОЛЖНЫ иметь атрибут* `targetNamespace` *с непустым и правильным значением, ЕСЛИ только элемент* `xsd:schema` *не имеет* `xsd:import` *и/или* `xsd:annotation` *как единственного дочернего элемента(ов).*

5.1.1.2.3 soapenc:Array

В секции 2.2 WSDL 1.1 рекомендации для объявления типа массива могут интерпретироваться различным образом, что приводит к проблемам интероперабельности. Однако имеются другие более четкие методы определения массивов.

R2110 В ОПИСАНИИ объявления массивов НЕ ДОЛЖНЫ расширять или ограничивать тип `soapenc:Array`*.*

R2111 В ОПИСАНИИ объявления массивов НЕ ДОЛЖНЫ использовать атрибут `wsdl:arrayType` *в объявлении типа.*

R2112 В ОПИСАНИИ элементы оболочки объявления массивов НЕ СЛЕДУЕТ именовать, используя шаблон ArrayOfXXX.

R2113 СООБЩЕНИЕ, содержащее сериализованные массивы, НЕ ДОЛЖНО расширять включать атрибут `soapenc:Array`*.*

Например:

НЕПРАВИЛЬНО:

```
Given the WSDL Description:
<xsd:element name="MyArray2" type="tns:MyArray2Type"/>
<xsd:complexType name="MyArray2Type"
```

```
 xmlns:soapenc="http://schemas.xmlsoap.org/soap/encoding/"
  xmlns:wsdl="http://schemas.xmlsoap.org/wsdl/" >
  <xsd:complexContent>
     <xsd:restriction base="soapenc:Array">
       <xsd:sequence>
          <xsd:element name="x" type="xsd:string"
           minOccurs="0" maxOccurs="unbounded"/>
       </xsd:sequence>
       <xsd:attribute ref="soapenc:arrayType"
        wsdl:arrayType="tns:MyArray2Type[]"/>
   </xsd:restriction>
 </xsd:complexContent>
</xsd:complexType>
The SOAP message would serialize as (omitting namespace declarations for clarity):
<MyArray2 soapenc:arrayType="tns:MyArray2Type[]" >
  <x>abcd</x>
  <x>efgh</x>
</MyArray2>
```

ПРАВИЛЬНО:

```
Given the WSDL Description:
<xsd:element name="MyArray1" type="tns:MyArray1Type"/>
<xsd:complexType name="MyArray1Type">
  <xsd:sequence>
   <xsd:element name="x" type="xsd:string"
    minOccurs="0" maxOccurs="unbounded"/>
  </xsd:sequence>
</xsd:complexType>
The SOAP message would serialize as (omitting namespace declarations for clarity):
<MyArray1>
  <x>abcd</x>
  <x>efgh</x>
</MyArray1>
```

5.1.1.2.4 Целевые пространства имен WSDL и определения схемы soapenc:Array

В секции 2.2 WSDL 1.1 рекомендации для объявления типа массива могут интерпретироваться различным образом, что приводит к проблемам интероперабельности. Однако имеются другие более четкие методы определения массивов.

R2110 *В ОПИСАНИИ объявления массивов НЕ ДОЛЖНЫ расширять или ограничивать тип* `soapenc:Array`.

R2111 *В ОПИСАНИИ объявления массивов НЕ ДОЛЖНЫ использовать атрибут* `wsdl:arrayType` *в объявлении типа.*

R2112 *В ОПИСАНИИ элементы оболочки объявления массивов НЕ СЛЕДУЕТ именовать, используя шаблон ArrayOfXXX.*

R2113 *СООБЩЕНИЕ, содержащее сериализованные массивы, НЕ ДОЛЖНО расширять включать атрибут* `soapenc:Array`.

Например:

НЕПРАВИЛЬНО:

```
Given the WSDL Description:
<xsd:element name="MyArray2" type="tns:MyArray2Type"/>
<xsd:complexType name="MyArray2Type"
 xmlns:soapenc="http://schemas.xmlsoap.org/soap/encoding/"
  xmlns:wsdl="http://schemas.xmlsoap.org/wsdl/" >
  <xsd:complexContent>
     <xsd:restriction base="soapenc:Array">
       <xsd:sequence>
          <xsd:element name="x" type="xsd:string"
           minOccurs="0" maxOccurs="unbounded"/>
       </xsd:sequence>
       <xsd:attribute ref="soapenc:arrayType"
        wsdl:arrayType="tns:MyArray2Type[]"/>
   </xsd:restriction>
 </xsd:complexContent>
</xsd:complexType>
The SOAP message would serialize as (omitting namespace declarations for clarity):
<MyArray2 soapenc:arrayType="tns:MyArray2Type[]" >
  <x>abcd</x>
  <x>efgh</x>
</MyArray2>
```

ПРАВИЛЬНО:

```
Given the WSDL Description:
<xsd:element name="MyArray1" type="tns:MyArray1Type"/>
<xsd:complexType name="MyArray1Type">
  <xsd:sequence>
   <xsd:element name="x" type="xsd:string"
    minOccurs="0" maxOccurs="unbounded"/>
  </xsd:sequence>
```

```
</xsd:complexType>
The SOAP message would serialize as (omitting namespace declarations for clarity):
<MyArray1>
  <x>abcd</x>
  <x>efgh</x>
</MyArray1>
```

Определенные в схемах имена и имена, указанные для определений WSDL, находятся в разных пространствах.

R2114 Целевое пространство определений WSDL и целевое пространство определений схемы в ОПИСАНИИ МОГУТ совпадать. WSDL20

5.1.1.3Сообщения

В данном разделе BasicProfile-1.0 используются следующие спецификации (некоторые их разделы):

- WSDL 1.1, Section 2.3

Элементы *wsdl:message* в WSDL 1.1 используются для представления абстрактных определений передаваемых данных. При этом элементы *wsdl:binding* используются для определения того, как эти абстрактные определения привязываются к транспортному формату. BasicProfile-1.0 формулирует следующие ограничения на элементы *wsdl:message* и на то, как элементы *wsdl:binding* могут использовать элемент(-ы) *wsdl:message*.

Для более компактного и ясного изложения этих ограничений далее в данной секции используются следующие определения.

«Rpc-литеральное связывание» это элемент *wsdl:binding*, все элементы-потомки *wsdl:operation* которого являются rpc-литеральными операциями.

«Rpc-литеральная операция» это дочерний *wsdl:operation*-элемент от *wsdl:binding*, все *soapbind:body*-элементы-потомки которого имеют атрибут *use* со значением "literal" и каждый из которых:

1. содержит атрибут стиля со значением «rpc» или

2. является потомком элемента *soapbind:binding* , содержащего атрибут стиля со значением «rpc», но сам не содержит атрибут стиля.

«Документ-литеральная операция» это дочерний *wsdl:operation*-элемент от *wsdl:binding*, все *soapbind:body*-элементы-потомки которого имеют атрибут *use* со значением "literal" и каждый из которых:

1. содержит атрибут стиля со значением «document» или

2. является потомком элемента *soapbind:binding* , содержащего атрибут стиля со значением «document», но сам не содержит атрибут стиля, или

3. является потомком элемента *soapbind:binding* , который не содержит атрибут стиля, и сам также не содержит атрибут стиля.

5.1.1.3.1 Связывание и части

Имеются различные понимания того, сколько разрешено или требуется элементов *wsdl:part* для rpc-литерального и документ-литерального связывания, и как они должны определяться.

R2201 Документ-литеральное связывание в ОПИСАНИИ ДОЛЖНО в каждом своем элементе(-ах) soapbind:body не более чем одну часть, перечисленную в атрибуте parts , если этот атрибут специфицирован.

R2210 Если документ-литеральное связывание в ОПИСАНИИ не содержит атрибута parts в элементе soapbind:body, соответствующее абстрактное wsdl:message ДОЛЖНО определять не более одной wsdl:part.

R2202 wsdl:binding в ОПИСАНИИ МОЖЕТ содержать элемент(-ы) soapbind:body, не содержащие частей из soap:Body.

R2203 Rpc-литеральное связывание в ОПИСАНИИ ДОЛЖНО ссылаться в своем элементе(-ах) soapbind:body только на те элементы *wsdl:part,* которые были определены с использованием атрибута *type*.

R2211 СООБЩЕНИЕ, описываемое с rpc-литеральным связыванием НЕ ДОЛЖНО иметь в ссылочной части атрибута xsi:nil со значением «1» или «true».

R2207 wsdl:message в ОПИСАНИИ МОЖЕТ содержать wsdl:part, использующий атрибут elements , только если на него нет ссылки из soapbind:body в rpc-литеральном связывании.

R2204 Документ-литеральное связывание в ОПИСАНИИ ДОЛЖНО *ссылаться в каждом своем soapbind:body элементе(-ах) только на те элементы wsdl:part, которые были определены с использованием атрибута element .*

R2208 *Связывание в ОПИСАНИИ* МОЖЕТ *содержать элемент(-ы)* `soapbind:header`, которые ссылаются на `wsdl:part` в том же `wsdl:message`, и на которые ссылается его элемент(-ы) `soapbind:body`.

Использование элементов `wsdl:message` с пустыми частями допускается при документных стилях для того чтобы разрешить операции, которые получают или принимают сообщения с пустым `soap:Body` . Использование элементов `wsdl:message` с пустыми частями допускается при rpc-стилях для того чтобы разрешить операции, которые не имеют параметров и\или не возвращают значение.

Для документ-литеральных связываний BasicProfile-1.0 требует, чтобы, по крайней мере, одна часть, абстрактно определенная с атрибутом `element`, была сериализированна в элементе `soap:Body.`

Когда элемент `wsdl:part` определяется с использованием атрибута `type`, транспортное представление этой части эквивалентно неявной (XML Schema) квалификации атрибутом `minOccurs` со значением «1», атрибутом `maxOccurs` со значением «1» и атрибутом `nillable` со значением "false".

5.1.1.3.2 Связывания и ошибки

Имеются различные понимания того, как могут определяться элемент `wsdl:part`, описвающие `soapbind:fault`, `soapbind:header` и `soapbind:headerfault` .

R2205 А `wsdl:binding` *в ОПИСАНИИ* ДОЛЖНО *ссылаться в каждом его элементе* `soapbind:header`, `soapbind:headerfault` *и* `soapbind:fault` *только на элемент(-ы)* `wsdl:part`, *который был определен с использованием атрибута* `element` .

Поскольку реакции на ошибки и заголовки не содержат параметров, `soapbind:fault`, `soapbind:header` и `soapbind:headerfault` предполагают, по WSDL 1.1, что значением атрибута `style` является "document". R2204 требует, чтобы все элементы `wsdl:part` с атрибутом `style`, имеющим значение "document", связанные с `soapbind:body`, были определены с использованием атрибута `element` . Это требование распространяется и на элементы `soapbind:fault`, `soapbind:header` и `soapbind:headerfault`.

5.1.1.3.3 Значения несвязанных элементов portType

WSDL 1.1 не вполне четко определяет, можно ли в *wsdl:binding* оставить неспецифицированными некоторые части значений, определяемых *wsdl:portType*.

R2209 *wsdl: binding* ***в*** *ОПИСАНИИ* СЛЕДУЕТ ***связывать каждую*** *wsdl:part* ***из*** *wsdl:message* ***в*** *wsdl:portType,* на которую он указывает, с одним из *soapbind:body*, *soapbind:header*, *soapbind:fault* or *soapbind:headerfault* .

PortType определяет абстрактный контракт между именованным набором операций и связанными с ними абстрактными сообщениями. Хотя и не декларируется обязательным, ожидается, что каждая часть абстрактных вводимых, выводимых или указывающих на ошибку сообщений, специфицированных в *portType*, связана с *soapbind:body* или *soapbind:header* (и т.д.), как это принято при использовании – связывания по определениям Секции 3 WSDL 1.1.

5.1.1.3.4 Объявление элементов частей

Примеры 4 и 5 Секции 3.1 WSDL 1.1 некорректно показывают использование типов XML Schema (например, "xsd:string"), как валидных значений атрибута *element* элемента *wsdl:part*.

R2206 *wsdl: message* ***в*** *ОПИСАНИИ, содержащем* *wsdl:part, использующую атрибут* *element, ДОЛЖНО ссылаться в этом атрибуте на глобальное определение элемента.*

Например:

НЕПРАВИЛЬНО:

```
<message name="GetTradePriceInput">
    <part name="tickerSymbol" element="xsd:string"/>
    <part name="time" element="xsd:timeInstant"/>
</message>
```

НЕПРАВИЛЬНО:

```
<message name="GetTradePriceInput">
    <part name="tickerSymbol" element="xsd:string"/>
</message>
```

ПРАВИЛЬНО:

```
<message name="GetTradePriceInput">
    <part name="body" element="tns:SubscribeToQuotes"/>
</message>
```

5.1.1.4Типы портов Сообщения

В данном разделе BasicProfile-1.0 используются следующие спецификации (некоторые их разделы):

- WSDL 1.1, Section 2.3

Элементы *wsdl:message* в WSDL 1.1 используются для представления абстрактных определений передаваемых данных. При этом элементы *wsdl:binding* используются для определения того, как эти абстрактные определения привязываются к транспортному формату. BasicProfile-1.0 формулирует следующие ограничения на элементы *wsdl:message* и на то, как элементы *wsdl:binding* могут использовать элемент(-ы) *wsdl:message*.

Для более компактного и ясного изложения этих ограничений далее в данной секции используются следующие определения.

«Rpc-литеральное связывание» это элемент *wsdl:binding*, все элементы-потомки *wsdl:operation* которого являются rpc-литеральными операциями.

«Rpc-литеральная операция» это дочерний *wsdl:operation*-элемент от *wsdl:binding*, все *soapbind:body*-элементы-потомки которого имеют атрибут *use* со значением "literal" и каждый из которых:

1. содержит атрибут стиля со значением «rpc» или
2. является потомком элемента *soapbind:binding* , содержащего атрибут стиля со значением «rpc», но сам не содержит атрибут стиля.

«Документ-литеральная операция» это дочерний *wsdl:operation*-элемент от *wsdl:binding*, все *soapbind:body*-элементы-потомки которого имеют атрибут *use* со значением "literal" и каждый из которых:

1. содержит атрибут стиля со значением «document» или
2. является потомком элемента *soapbind:binding* , содержащего атрибут стиля со значением «document», но сам не содержит атрибут стиля, или
3. является потомком элемента *soapbind:binding* , который не содержит атрибут стиля, и сам также не содержит атрибут стиля.

В данном разделе BasicProfile-1.0 используются следующие спецификации (некоторые их разделы):

- WSDL 1.1, Section 2.4

В WSDL 1.1 элементы `wsdl:portType` используются для группировки набора абстрактных операций. BasicProfile-1.0 накладывает следующие ограничения на элемент (-ы) `wsdl:portType`.

5.1.1.4.1 Порядок элементов частей

Допустимость использования `parameterOrder` помогает генераторам кода в отображении между сигнатурами и передаваемыми сообщениями.

R2301 *Порядок элементов в `soap:body` СООБЩЕНИЯ ДОЛЖЕН совпадать с порядком `wsdl:part`s в описывающем его `wsdl:message`.*

R2302 *ОПИСАНИЕ МОЖЕТ использовать атрибут `parameterOrder` элемента `wsdl:operation` для указания сигнатур возвращаемого значения и метода, как подсказку для генераторов кода.*

5.1.1.4.2 Допустимые операции

Операции Solicit-Response (требование ответа) и Notification (уведомление) нечетко определены в WSDL 1.1, и кроме того, WSDL 1.1 не определяет их связывания.

R2303 *ОПИСАНИЕ НЕ ДОЛЖНО использовать типы операций Solicit-Response и Notification в определении `wsdl:portType`.*

5.1.1.4.3 Различимость операций

BasicProfile-1.0 не допускает неоднозначности имен операций в `wsdl:portType`.

R2304 *`wsdl:portType` в ОПИСАНИИ ДОЛЖЕН иметь операции с различными значениями атрибута `name`.*

5.1.1.4.4 Конструкция атрибута parameterOrder

WSDL 1.1 нечетко определяет, какова конструкция атрибута `parameterOrder` в `wsdl:portType`.

R2305 *`wsdl:portType` в ОПИСАНИИ ДОЛЖЕН конструироваться так, чтобы атрибут `parameterOrder`, если он присутствует, опускал не более одной `wsdl:part` в выдаваемом сообщении.*

Если некоторая *wsdl:part* в выдаваемом сообщении отсутствует в списке *wsdl:part*, являющимся значением атрибута *parameterOrder*, то эта единственная опущенная *wsdl:part* является возвращаемым значением. Не имеется никаких ограничений на тип возвращаемого значения. Если нет ни одной отсутствующей части, то нет возвращаемого значения.

5.1.1.4.5 Взаимоисключение атрибутов типа и элемента

WSDL 1.1 нечетко указывает, что атрибуты *type* и *element* не могут быть одновременно специфицированы в определении *wsdl:part* в *wsdl:message*.

R2306 *wsdl: message* ***в ОПИСАНИИ НЕ ДОЛЖЕН одновременно иметь атрибуты*** *type* ***и*** *element* ***в некоторой*** *wsdl:part.*

5.1.1.5Связывания

В данном разделе BasicProfile-1.0 используются следующие спецификации (некоторые их разделы):

- WSDL 1.1, Section 2.5

В WSDL 1.1 элемент *wsdl: binding* указывает конкретный протокол и спецификации формата данных для операций и сообщений, определяемых некоторым *wsdl:portType*. BasicProfile-1.0 накладывает следующие ограничения на спецификации связывания:

5.1.1.5.1 Использование SOAP-связывания

BasicProfile-1.0 ограничивает выбор вариантов связывания лишь хорошо определенным и наиболее широко используемым SOAP-связыванием, не разрешая использование связываний MIME и HTTP GET/POST.

R2401 *Элемент* *wsdl: binding* ***в ОПИСАНИИ ДОЛЖЕН использовать*** WSDL SOAP-*связывание, как это определено в Секции 3 WSDL 1.1.*

Заметим, что тем самым формулируется требование к соответствующим Профилю элементам *wsdl: binding* , а не к описанию в целом, в частности, это не запрещает WSDL-документы с элементами *wsdl: binding*, не соответствующим Профилю.

5.1.1.6SOAP-связывание

В данном разделе BasicProfile-1.0 используются следующие спецификации (некоторые их разделы):

- WSDL 1.1, Section 3.0

WSDL 1.1 определяет связывание для конечных точек SOAP 1.1. BasicProfile-1.0 утверждает обязательность использования SOAP-связывания как оно определено в WSDL 1.1 и формулирует следующие ограничения на его использование:

5.1.1.6.1 Спецификация атрибута transport

Имеются расхождения в спецификациях WSDL 1.1 и WSDL 1.1 Schema в отношении атрибута transport. Спецификации WSDL 1.1 объявляют его обязательным, в то время как схема разрешает его необязательность.

R2701 Элемент `wsdl:binding` *в ОПИСАНИИ ДОЛЖЕН быть построен так, чтобы его дочерний элемент* `soapbind:binding` *содержал атрибут* `transport`.

5.1.1.6.2 HTTP-транспорт

BasicProfile-1.0 в качестве транспортного протокола определяет HTTP.

R2702Элемент `wsdl: binding` *в ОПИСАНИИ ДОЛЖЕН быть специфицировать транспортный протокол HTTP с SOAP-связывание, т.е. атрибут* `transport` *его потомка* `soapbind:binding` *ДОЛЖЕН иметь значение* "http://schemas.xmlsoap.org/soap/http".

Заметим, что данное требование не исключает использования HTTPS, см. R5000.

5.1.1.6.3 Согласованность атрибута style

Стиль взаимодействия (`style`), "document" или "rpc", указывается на уровне `wsdl:operation` , позволяя иметь `wsdl:binding`s, у которых разные значения `style`. Это приводит к проблемам интероперабельности.

R2705 `wsdl: binding` *в ОПИСАНИИ ДОЛЖЕН использовать либо rpc-литеральное связывание, либо документ- литеральное связывание.*

5.1.1.6.4 Кодирование и атрибут use

BasicProfile-1.0 запрещает использование кодирования, включая SOAP-кодирование.

R2706 `wsdl:binding` *в ОПИСАНИИ ДОЛЖЕН использовать значение "literal" для атрибута* `use` *во всех элементах* `soapbind:body`, `soapbind:fault`, `soapbind:header` *и* `soapbind:headerfault`.

5.1.1.6.5 Значение по умолчанию атрибута use

Имеются расхождения в спецификациях WSDL 1.1 и WSDL 1.1 Schema в отношении того, является ли атрибут `use` необязательным в `soapbind:body`, `soapbind:header`, и `soapbind:headerfault`, и если так, то что означает его отсутствие.

R2707 `wsdl:binding` *в ОПИСАНИИ, где содержится один или более элементов* `soapbind:body`, `soapbind:fault`, `soapbind:header` *или* `soapbind:headerfault`, *не специфицирующих атрибут* `use`, *должно интерпретироваться так, словно в каждом случае было специфицировано значение* "literal".

5.1.1.6.6 Множественные связывания для элементов portType

BasicProfile-1.0 явно разрешает множественные связывания для одного и того же portType.

R2709 `wsdl: portType` *в ОПИСАНИИ МОЖЕТ иметь несколько (или ни одного) ссылающихся на него* `wsdl:binding`, *определенных в одном и том же или в разных WSDL –документах.*

5.1.1.6.7 Wire-сигнатуры для операций

Конечная точка, поддерживающая несколько операций, должна однозначно определять вызываемую операцию по принятому входному сообщению. Это возможно лишь в том случае, если все операции, специфицированные в `wsdl:binding`, связанном с данной конечной точкой, имеют различные wire-сигнатуры.

R2710 Операции в `wsdl:binding` *в ОПИСАНИИ ДОЛЖНЫ иметь различные wire-сигнатуры.*

BasicProfile-1.0 определяет «wire-сигнатуру» некоторой операции в `wsdl:binding` как полностью квалифицированное имя дочернего элемента `soap:Body` входного SOAP-сообщения, которое оно описывает. В случае пустого `soap:Body` это имя представляет собой пустую строку.

В случае rpc-литерального связывания имя операции используется как оболочка для обращений к элементу-части. В случае документ-литерального связывания, когда оболочка с именем операции отсутствует, сигнатуры сообщения должны быть построены корректно в соответствии с указанным выше требованием.

5.1.1.6.8 Множественные порты одной конечной точки

В случае, когда входные сообщения, предназначенные для двух различных `wsdl:port` одной и той же конечной точки, неразличимы в транспорте, невозможно установить вызываемый ими `wsdl:port`. Это может вызвать проблемы интероперабельности. Однако могут встречаться ситуации (например, версионность SOAP, версионность приложения, соответствие различным профилям), когда желательно располагать более чем один порт в одной конечной точке, поэтому данный Профиль допускает это.

R2711 ОПИСАНИЮ НЕ СЛЕДУЕТ иметь более чем один `wsdl:port` *с тем же самым значением атрибута* `location` *элемента* `soapbind:address`.

5.1.1.6.9 Дочерний элемент при документ-литеральном связывании

В WSDL 1.1 не вполне ясно, что представляет собой дочерний элемент в `soap:Body` при документ-литеральном связывании.

R2712 Документ-литеральное связывание ДОЛЖНО быть представлено в транспорте как СООБЩЕНИЕ с `soap:Body` *, дочерний элемент которого есть экземпляр глобальной декларации элемента, указываемой соответствующей частью* `wsdl:message`.

5.1.1.6.10 Однонаправленные операции

Имеются различные интерпретации того, как используется HTTP при однонаправленных операциях.

R2714 Для однонаправленных операций ЭКЗЕМПЛЯР НЕ ДОЛЖЕН возвращать HTTP-ответ, содержащий SOAP-конверт, т.е. объект-тело HTTP-ответа должно быть пустым.

R2750 ПОЛУЧАТЕЛЬ ДОЛЖЕН игнорировать SOAP-ответ, получаемый в ответ на однонаправленную операцию.

R2727 Для однонаправленных операций ПОЛУЧАТЕЛЬ НЕ ДОЛЖЕН интерпретировать успешный код (например, 2xx) HTTP-ответа в смысле валидности сообщения или будто бы он будет его обрабатывать.

Однонаправленные операции не выдают SOAP-ответов. Поэтому BasicProfile-1.0 запрещает посылку SOAP-конверта в ответ на однонаправленную операцию. Это означает, что пересылка однонаправленных операций не может вызвать ответов или ошибок уровня обработки. Например, HTTP-ответ "500 Internal Server Error", включающий SOAP-сообщение, содержащее SOAP-элемент ошибки, не может быть возвращено.

HTTP-ответ на однонаправленную операцию указывает успешность или неудачу передачи сообщения. Основываясь на семантике различных кодов состояния ответа, поддерживаемых HTTP-протоколом, BasicProfile-1.0 указывает, что отправитель должен предпочтительно ожидать коды «200» и «202», означающие, что однонаправленное сообщение было получено. Успешная передача не означает, что обрабатывающий слой SOAP или логика приложения имели возможность проверить правильность этого сообщения или пытались его обработать.

Несмотря на тот факт, что HTTP 1.1 указывает различные толкования ответных кодов состояния «200» и «202», в контексте BasicProfile-1.0 они должны интерпретироваться одинаково инициатором запроса. BasicProfile-1.0 допускает оба эти кода, т.к. некоторые реализации SOAP недостаточно контролируют реализацию протокола HTTP и не могут отвечать за то, какой из этих кодов будет послан.

5.1.1.6.11 Пространства имен для элементов soapbind

Не вполне ясно, какое пространство имен связывается с дочерними элементами различных потомков `soap:Envelope`, что приводит к проблемам интероперабельности. BasicProfile-1.0 уточняет эту ситуацию

R2716 Документ-литеральное связывание в ОПИСАНИИ НЕ ДОЛЖНО иметь атрибута `namespace` *, указанного для содержащихся в нем элементов* `soapbind:body`, `soapbind:header`, `soapbind:headerfault` и `soapbind:fault`.

R2717 Rpc-литеральное связывание в ОПИСАНИИ ДОЛЖНО иметь атрибут `namespace` *, указанный для содержащихся в нем элементов* `soapbind:body`*, значением которого должен быть абсолютный URI.*

R2726 Rpc-литеральное связывание в ОПИСАНИИ НЕ ДОЛЖНО иметь атрибута `namespace`*, указанного для содержащихся в нем элементов* `soapbind:header`, `soapbind:headerfault` и `soapbind:fault`.

В документ-литеральном SOAP-связывании элемент-последовательность дочерний от `soap:Body` получает свое пространство имен через targetNamespace схемы, определяющей этот элемент. Напротив, в rpc-литеральном SOAP-связывании элемент-последовательность дочерний от `soap:Body` состоит из элемента-оболочки, пространство имен которого есть значение атрибута `namespace` элемента `soapbind:body` , и локальное имя которого есть либо имя операции, либо имя операции с суффиксом "Response". Атрибут `namespace` здесь является обязательным для обеспечения указания пространства имен для потомков элемента `soapbind:body`.

5.1.1.6.12 Согласованность элементов portType и связывания

WSDL-описание должно быть согласованным на уровнях `wsdl:portType` и `wsdl:binding`

R2718 `wsdl:binding` *в ОПИСАНИИ ДОЛЖНО иметь тот же самый набор* `wsdl:operation`*, что и в* `wsdl:portType`*, на который оно ссылается.*

5.1.1.6.13 Описание элементов headerfault

Имеется несоответствие между спецификациями WSDL 1.1 и WSDL 1.1 -схемы относительно `soapbind:headerfault`.

R2719 `wsdl:binding` *в ОПИСАНИИ МОЖЕТ содержать элементы* `soapbind:headerfault`*, если в заголовке не обнаружено ошибок.*

Схема WSDL 1.1 утверждает обязательность элемента `soapbind:headerfault` для элементов `wsdl:input` и `wsdl: output` операции, в то время как спецификации WSDL 1.1 определяют здесь необязательность. Эти последние принимаются как правильные.

5.1.1.6.14 Перечисление ошибок

Описание Web-сервиса должно включать все ошибки, известные на момент определения сервиса. Должна также иметься возможность генерации новых ошибок, не распознанных при определении сервиса.

R2740 `wsdl:binding` *в ОПИСАНИИ СЛЕДУЕТ содержать* `soapbind:fault` *для описания каждой из известных ошибок.*

R2741 `wsdl:binding` *в ОПИСАНИИ СЛЕДУЕТ содержать* `soapbind:headerfault` *для описания каждой из известных ошибок заголовка.*

R2742 СООБЩЕНИЕ МОЖЕТ содержать вхождение о деталях ошибки в SOAP-ошибке, не описанной в элементе `wsdl:fault` *соответствующего WSDL-описания.*

R2743 СООБЩЕНИЕ МОЖЕТ содержать подробные сведения об ошибке обработки заголовка в SOAP-блоке заголовка, не описанной в элементе `wsdl:headerfault` *соответствующего WSDL-описания.*

5.1.1.6.15 Типы и имена элементов of SOAP-связывания

Имеется несоответствие между спецификациями WSDL 1.1 и WSDL 1.1 -схемы относительно имени и типа атрибута `soapbind:header` и элементов `soapbind:headerfault`.

R2720 `wsdl:binding` *в ОПИСАНИИ ДОЛЖЕН использовать имя атрибута* `part` *со схемным типом* "NMTOKEN" *для всех содержащихся в нем элементов* `soapbind:header` *и* `soapbind:headerfault`.

R2749 `wsdl:binding` *в ОПИСАНИИ НЕ ДОЛЖЕН использовать имя атрибута* `parts` *для содержащихся в нем элементов* `soapbind:header` *и* `soapbind:headerfault`.

WSDL Schema определяет имя атрибута "parts" и его тип как "NMTOKENS". Подход схемы в этом случае некорректен, т.к. каждый элемент `soapbind:header` и `soapbind:headerfault` ссылаются на один и тот же `wsdl:part`.

Например:

ПРАВИЛЬНО:

```
<binding name="StockQuoteSoap" type="tns:StockQuotePortType">
  <soapbind:binding style="document"
                transport="http://schemas.xmlsoap.org/soap/http"/>
```

```
    <operation name="SubscribeToQuotes">
      <input message="tns:SubscribeToQuotes">
        <soapbind:body parts="body" use="literal"/>
        <soapbind:header message="tns:SubscribeToQuotes"
                        part="subscribeheader" use="literal"/>
     </input>
   </operation>
</binding>
```

5.1.1.6.16 Атрибут name в ошибках Типы и имена элементов of SOAP-связывания

Имеется несоответствие между спецификациями WSDL 1.1 и WSDL 1.1 -схемы относительно имени и типа атрибута *soapbind:header* и элементов *soapbind:headerfault*.

R2720 *wsdl:binding* ***в ОПИСАНИИ ДОЛЖЕН использовать имя атрибута*** *part* ***со схемным типом*** "NMTOKEN" ***для всех содержащихся в нем элементов*** *soapbind:header* ***и*** *soapbind:headerfault*.

R2749 *wsdl:binding* ***в ОПИСАНИИ НЕ ДОЛЖЕН использовать имя атрибута*** *parts* ***для содержащихся в нем элементов*** *soapbind:header* ***и*** *soapbind:headerfault*.

WSDL Schema определяет имя атрибута "parts" и его тип как "NMTOKENS". Подход схемы в этом случае некорректен, т.к. каждый элемент *soapbind:header* и *soapbind:headerfault* ссылаются на один и тот же *wsdl:part*.

Например:

ПРАВИЛЬНО:

```
<binding name="StockQuoteSoap" type="tns:StockQuotePortType">
  <soapbind:binding style="document"
                transport="http://schemas.xmlsoap.org/soap/http"/>
    <operation name="SubscribeToQuotes">
      <input message="tns:SubscribeToQuotes">
        <soapbind:body parts="body" use="literal"/>
        <soapbind:header message="tns:SubscribeToQuotes"
                        part="subscribeheader" use="literal"/>
     </input>
```

```
   </operation>
</binding>
```

Имеется несоответствие между спецификациями WSDL 1.1 и WSDL 1.1-схемы, в которых не определяется использование атрибута `name`.

R2721 `wsdl:binding` *в ОПИСАНИИ ДОЛЖЕН иметь атрибут* `name` *специфицированным для всех содержащихся в нем элементов* `soapbind:fault`.

R2754 В ОПИСАНИИ значение атрибута `name` *элемента* `soapbind:fault` *ДОЛЖНО соответствовать значение атрибута* `name` *его головного элемента* `wsdl:fault`.

5.1.1.6.17 Отсутствие атрибута use

Имеется несоответствие между спецификациями WSDL 1.1 и WSDL 1.1-схемы касающихся атрибута `use`.

R2722 `wsdl:binding` *в ОПИСАНИИ МОЖЕТ иметь атрибут* `use` *специфицированным для всех содержащихся в нем элементов* `soapbind:fault`.

R2723 Если в `wsdl:binding` *в ОПИСАНИИ МОЖЕТ иметь атрибут* `use` *присутствует для содержащегося в нем элемента* `soapbind:fault` *, его значение ДОЛЖНО быть* "literal".

R2728 `wsdl:binding` *в ОПИСАНИИ, где опущен атрибут* `use` *для содержащегося в нем элемента* `soapbind:fault` *, ДОЛЖЕН интерпретироваться так, словно был специфицирован* `use`="literal".

В Секции 3.5 WSDL 1.1 указывается, что атрибут `use` для `soapbind:fault` является обязательным, в то время как в схеме использование атрибута `use` необязательно. BasicProfile-1.0 определяет его необязательным для совместимости с `soapbind:body`.

В связи с необязательностью атрибута `use` BasicProfile-1.0 определяет его дефолтное значение.

Наконец, для обеспечения своей непротиворечивости, BasicProfile-1.0 определяет для атрибута `use` единственное допустимое значение "literal".

5.1.1.6.18 Согласованность сообщений и описаний

Это требование специфицирует, что когда экземпляр получает сообщение, не согласующееся с WSDL-описанием, должна генерироваться ошибка, если только экземпляр не берет на себя, несмотря на это, обработку данного сообщения.

Как указано в SOAP-модели обработки, (a) при некорректности элемента "Envelope" должен выдаваться код ошибки "VersionMismatch", (b) ошибка "MustUnderstand" должна выдаваться в случае, когда экземпляр не понимает SOAP-блока заголовка со значением «1» атрибута `soap:mustUnderstand`. Во всех остальных случаях несоответствия сообщения WSDL-описанию должна выдаваться ошибка с кодом "Client".

R2724 Если ЭКЗЕМПЛЯР получает сообщение, не согласующееся с WSDL-описанием, ему СЛЕДУЕТ выдать `soap:Fault` *с кодом "Client", кроме случаев выдачи ошибок* "MustUnderstand" *или* "VersionMismatch".

R2725 Если ЭКЗЕМПЛЯР получает сообщение, не согласующееся с WSDL-описанием, он ДОЛЖЕН проверять состояния ошибок "Client","MustUnderstand"*,* "VersionMismatch" *и* "Client*" , именно в этом порядке.*

5.1.1.6.19 Оболочки ответов

Секция 3.5 WSDL 1.1 может интерпретироваться так, что элемент оболочки RPC-ответа должен именоваться идентично имени `wsdl:operation`.

R2729 СООБЩЕНИЕ, описанное с rpc-литеральным связыванием и являющееся ответным сообщением, ДОЛЖНО иметь элемент оболочки, имя которого соответствует имени `wsdl:operation` *с суффиксом "Response".*

5.1.1.6.20 Пространство имен для аксессоров частей

Для rpc-литеральных SOAP-сообщений в WSDL 1.1 неясно, какому пространству имен принадлежат (если принадлежат какому-то) аксессор-элементы для параметров и возвращаемого значения. Различия в интерпретации приводят к различиям выбора, что создает проблемы интероперабельности.

R2735 СООБЩЕНИЕ, описанное с rpc-литеральным связыванием НЕ ДОЛЖНО размещать элементы аксессоров частей в каком-либо пространстве имен.

Формулировка единственной альтернативы критична для обеспечения интероперабельности. BasicProfile-1.0 определяет, что для элементов аксессоров частей

не следует указывать пространства имен, т.к. это более простое решение, покрывающее все случаи и не вызывающее логических противоречий.

5.1.1.6.21 Пространства имен для наследников аксессоров частей

Для rpc-литеральных SOAP-сообщений в WSDL 1.1 неясно, каково корректное уточнение пространства имен для элементов-наследников аксессор-элементов частей, когда соответствующие абстрактные части определены с типами из пространства имен, отличного от *targetNamespace* WSDL-описания для абстрактных частей.

R2737 СООБЩЕНИЕ, описанное с rpc-литеральным связыванием ДОЛЖНО квалифицировать потомков элементов-аксессоров частей для параметров и возвращаемого значения указанием пространства имен из targetNamespace, в котором определены их типы.

WSDL 1.1 в Секции 3.5 утверждает: «Имена, типы и значения атрибута пространства имен являются входными значениями кодирования, хотя атрибут пространства имен применяется только к содержанию, не определяемому явно абстрактными типами.

Однако здесь нет явного утверждения, что содержание элемента и атрибута абстрактных типов (complexType) квалифицируется указанием пространства имен из targetNamespace, в котором эти элементы и атрибуты были определены. Предполагалось, что WSDL 1.1 будет в основном соответствовать XML Schema. Следовательно, реализации должны соблюдать правила XML Schema. Если complexType, определенный в *targetNamespace* «A», был импортирован и указан в объявлении элемента в схеме с *targetNamespace* «B», содержание элемента и атрибута элементов-наследников этого complexType должно быть квалифицировано пространством имен «A», а сам элемент должен быть квалифицирован пространством имен «B».

Например:

ПРАВИЛЬНО:

```
Приводится WSDL, который определяет некоторую схему в пространстве имен
"http://example.org/foo/"  в секции wsdl:types содержащейся в wsdl:definitions ,
имеющим атрибут targetNamespace со значением "http://example.org/bar/" (т.е.
имеющий объявление типа в одном пространстве имен, а содержащий элемент – в
другом);
<definitions xmlns="http://schemas.xmlsoap.org/wsdl/"
xmlns:soap="http://schemas.xmlsoap.org/wsdl/soap/"
xmlns:soapbind="http://schemas.xmlsoap.org/wsdl/soap/"
```

```
xmlns:http="http://schemas.xmlsoap.org/wsdl/http/"
xmlns:xsd="http://www.w3.org/2001/XMLSchema"
xmlns:bar="http://example.org/bar/"
targetNamespace="http://example.org/bar/"
xmlns:foo="http://example.org/foo/">
<types>
   <xsd:schema targetNamespace="http://example.org/foo/"
       xmlns:tns="http://example.org/foo/"
       xmlns:xsd="http://www.w3.org/2001/XMLSchema"
       elementFormDefault="qualified"
       attributeFormDefault="unqualified">
       <xsd:complexType name="fooType">
          <xsd:sequence>
             <xsd:element ref="tns:bar"/>
             <xsd:element ref="tns:baf"/>
          </xsd:sequence>
       </xsd:complexType>
       <xsd:element name="bar" type="xsd:string"/>
       <xsd:element name="baf" type="xsd:integer"/>
   </xsd:schema>
</types>
<message name="BarMsg">
   <part name="BarAccessor" type="foo:fooType"/>
</message>
<portType name="BarPortType">
   <operation name="BarOperation">
     <input message="bar:BarMsg"/>
   </operation>
</portType>
<binding name="BarSOAPBinding" type="bar:BarPortType">
   <soapbind:binding
    transport="http://schemas.xmlsoap.org/soap/http/"
    style="rpc"/>
   <operation name="BarOperation">
     <input message="bar:BarMsg">
       <soapbind:body use="literal" namespace="http://example.org/bar/"/>
     </input>
   </operation>
</binding>
<service name="serviceName">
  <port name="BarSOAPPort" binding="bar:BarSOAPBinding">
    <soapbind:address location="http://example.org/myBarSOAPPort"/>
  </port>
```

```
</service>
</definitions>
Результирующее SOAP-сообщение для BarOperation:
<s:Envelope xmlns:s="http://schemas.xmlsoap.org/soap/envelope/"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xmlns:xsd="http://www.w3.org/2001/XMLSchema"
xmlns:foo="http://example.org/foo/">
  <s:Header/>
    <s:Body>
      <m:BarOperation xmlns:m="http://example.org/bar/">
         <BarAccessor>
            <foo:bar>String</foo:bar>
            <foo:baf>0</foo:baf>
         </BarAccessor>
      </m:BarOperation>
    </s:Body>
</s:Envelope>
```

5.1.1.6.22 Обязательность заголовков Пространства имен для наследников аксессоров частей

Для rpc-литеральных SOAP-сообщений в WSDL 1.1 неясно, каково корректное уточнение пространства имен для элементов-наследников аксессор-элементов частей, когда соответствующие абстрактные части определены с типами из пространства имен, отличного от *targetNamespace* WSDL-описания для абстрактных частей.

R2737 СООБЩЕНИЕ, описанное с rpc-литеральным связыванием ДОЛЖНО квалифицировать потомков элементов-аксессоров частей для параметров и возвращаемого значения указанием пространства имен из targetNamespace, в котором определены их типы.

WSDL 1.1 в Секции 3.5 утверждает: «Имена, типы и значения атрибута пространства имен являются входными значениями кодирования, хотя атрибут пространства имен применяется только к содержанию, не определяемому явно абстрактными типами.

Однако здесь нет явного утверждения, что содержание элемента и атрибута абстрактных типов (complexType) квалифицируется указанием пространства имен из targetNamespace, в котором эти элементы и атрибуты были определены. Предполагалось, что WSDL 1.1 будет в основном соответствовать XML Schema. Следовательно, реализации должны соблюдать правила XML Schema. Если complexType,

определенный в *targetNamespace* «A», был импортирован и указан в объявлении элемента в схеме с *targetNamespace* «B», содержание элемента и атрибута элементов-наследников этого complexType должно быть квалифицировано пространством имен «A», а сам элемент должен быть квалифицирован пространством имен «B».

Например:

ПРАВИЛЬНО:

Приводится WSDL, который определяет некоторую схему в пространстве имен "http://example.org/foo/" в секции *wsdl:types* содержащейся в *wsdl:definitions*, имеющим атрибут *targetNamespace* со значением "http://example.org/bar/" (т.е. имеющий объявление типа в одном пространстве имен, а содержащий элемент – в другом);

```
<definitions xmlns="http://schemas.xmlsoap.org/wsdl/"
xmlns:soap="http://schemas.xmlsoap.org/wsdl/soap/"
xmlns:soapbind="http://schemas.xmlsoap.org/wsdl/soap/"
xmlns:http="http://schemas.xmlsoap.org/wsdl/http/"
xmlns:xsd="http://www.w3.org/2001/XMLSchema"
xmlns:bar="http://example.org/bar/"
targetNamespace="http://example.org/bar/"
xmlns:foo="http://example.org/foo/">
<types>
   <xsd:schema targetNamespace="http://example.org/foo/"
       xmlns:tns="http://example.org/foo/"
       xmlns:xsd="http://www.w3.org/2001/XMLSchema"
       elementFormDefault="qualified"
       attributeFormDefault="unqualified">
       <xsd:complexType name="fooType">
          <xsd:sequence>
             <xsd:element ref="tns:bar"/>
             <xsd:element ref="tns:baf"/>
          </xsd:sequence>
       </xsd:complexType>
       <xsd:element name="bar" type="xsd:string"/>
       <xsd:element name="baf" type="xsd:integer"/>
   </xsd:schema>
</types>
<message name="BarMsg">
   <part name="BarAccessor" type="foo:fooType"/>
</message>
<portType name="BarPortType">
   <operation name="BarOperation">
     <input message="bar:BarMsg"/>
```

```
   </operation>
</portType>
<binding name="BarSOAPBinding" type="bar:BarPortType">
   <soapbind:binding
    transport="http://schemas.xmlsoap.org/soap/http/"
    style="rpc"/>
   <operation name="BarOperation">
     <input message="bar:BarMsg">
       <soapbind:body use="literal" namespace="http://example.org/bar/"/>
     </input>
   </operation>
</binding>
<service name="serviceName">
  <port name="BarSOAPPort" binding="bar:BarSOAPBinding">
    <soapbind:address location="http://example.org/myBarSOAPPort"/>
  </port>
</service>
</definitions>
Результирующее SOAP-сообщение для BarOperation:
<s:Envelope xmlns:s="http://schemas.xmlsoap.org/soap/envelope/"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xmlns:xsd="http://www.w3.org/2001/XMLSchema"
xmlns:foo="http://example.org/foo/">
  <s:Header/>
    <s:Body>
      <m:BarOperation xmlns:m="http://example.org/bar/">
         <BarAccessor>
            <foo:bar>String</foo:bar>
            <foo:baf>0</foo:baf>
         </BarAccessor>
      </m:BarOperation>
    </s:Body>
</s:Envelope>
```

WSDL 1.1 неясно указывает, должны ли все *soapbind:header* специфицированные для элементов *wsdl:input* или *wsdl:output* в секции SOAP-связывания WSDL-описания быть включены в результирующие SOAP-сообщения при их передаче. BasicProfile-1.0 определяет все эти заголовки как обязательные, т.к. в WSDL 1.1 отсутствует возможность маркировать заголовок как необязательный.

R2738 СООБЩЕНИЕ ДОЛЖНО включать все специфицированные специфицированные для wsdl:input или wsdl:output в `wsdl:operation` *в описывающем его* `wsdl:binding`.

5.1.1.6.23 Допустимость неописанных заголовков

Заголовки являются средством расширения SOAP. По разным причинам может понадобиться включить не определенные в WSDL-описании заголовки в SOAP-сообщения.

R2739 СООБЩЕНИЕ МОЖЕТ включать блоки заголовков, не определенные в описывающем его `wsdl:binding`.

R2753 СООБЩЕНИЕ, содержащее блоки заголовков, не определенные в соответствующем `wsdl:binding`., *МОЖЕТ для таких SOAP-блоков заголовков иметь атрибут* `mustUnderstand` *со значением «1».*

5.1.1.6.24 Порядок заголовков

Не требуется соответствия между порядком `soapbind:header` в описании и порядком SOAP-блоков заголовков в сообщении. Более того, в сообщении может присутствовать более одного экземпляра каждого специфицированного SOAP-блока заголовка.

R2751 Порядок элементов `soapbind:header` *в секциях* `soapbind:binding` *ОПИСАНИЯ ДОЛЖЕН приниматься как независимый от порядка SOAP-блокрв заголовков в сообщении.*

R2752 СООБЩЕНИЕ МОЖЕТ содержать более чем один экземпляр каждого SOAP-блока заголовка для каждого элемента `soapbind:header` *в соответствующем потомке* `soapbind:binding` *соответствующего описания.*

5.1.1.6.25 Описание SOAPAction

Тестирование интероперабельности показало, что требование указания значения поля `SOAPAction` HTTP-заголовка улучшает интероперабельность реализаций. Хотя HTTP и разрешает отсутствие указания значений полей заголовка, некоторые реализации требуют указания этих значений.

R2744 СООБЩЕНИЕ-запрос HTTP ДОЛЖНО содержать поле SOAPAction HTTP-заголовка с указанием значения равного значению атрибута `soapAction` *в* `soapbind:operation`*, если оно присутствует в соответствующем WSDL описании.*

R2745 СООБЩЕНИЕ-запрос HTTP ДОЛЖНО содержать поле SOAPAction HTTP-заголовка с указанием пустого единичного значения равного значению атрибута `soapAction` *в* `soapbind:operation`*, если в соответствующем WSDL описании* `soapAction` *либо отсутствует в* `soapbind:operation`*, либо присутствует с пустой строкой в качестве значения.*

Более детальное обсуждение `SOAPAction` см. в R1119 и соответствующих требованиях.

Например:

ПРАВИЛЬНО:

WSDL-описание, имеющее:

```
<soapbind:operation soapAction="foo" />
```

приводит к сообщению с соответствующим SOAPAction HTTP-полем заголовка в виде:

```
SOAPAction: "foo"
```

ПРАВИЛЬНО:

WSDL-описание, имеющее:`<soapbind:operation />`

или

```
<soapbind:operation soapAction="" />
```

приводит к сообщению с соответствующим SOAPAction HTTP-полем заголовка в виде:

```
SOAPAction: ""
```

5.1.1.6.26 Расширения SOAP-связывания

Атрибут `wsdl:required` часто неправильно трактуется, и иногда использовался авторами WSDL для указания необязательности `soapbind:header`. По спецификациям WSDL 1.1 атрибут `wsdl:required` представляет собой средство расширения для WSDL-процессоров. Он позволяет элегантно вводить новые элементы-расширения WSDL. Атрибут `wsdl:required` предназначен для указания процессору WSDL, следует ли ему распознавать и понимать элемент-расширение для правильной обработки WSDL-описания. Он не предназначался для указания условности или необязательности некоторой конструкции, включаемой в сообщения. Например, атрибут `wsdl:required` со значением "false" для элемента `soapbind:header` не должен интерпретироваться в смысле указания WSDL-процессору, что описываемый блок SOAP-заголовка является

условным или необязательным в сообщениях, генерируемых по WSDL-описанию. Его следует интерпретировать как «для того чтобы послать в конечный узел сообщение, которое имеет в своем описании элемент *soapbind:header*, WSDL-процессор ДОЛЖЕН понимать семантику, представляемую элементом *soapbind:header*».

Дефолтным значением атрибута *wsdl:required* для элементов-расширений WSDL 1.1 SOAP-связывания является "false". Большинство WSDL-описаний на практике не указывают атрибут *wsdl:required* для элементов-расширений SOAP-связывания, что может интерпретироваться WSDL-процессорами как указание на то, что эти элементы-расширения могут игнорироваться. BasicProfile-1.0 требует, чтобы все WSDL SOAP 1.1-расширения понимались и обрабатывались получателем, независимо от присутствия или значения атрибута *wsdl:required* для элемента-расширения.

R2747 ПОЛУЧАТЕЛЬ ДОЛЖЕН понимать и обрабатывать все элементы-расширения WSDL 1.1 SOAP-связывания, независимо от присутствия или значения атрибута wsdl:required для элемента-расширения.

R2748 ПОЛУЧАТЕЛЬ НЕ ДОЛЖЕН интерпретировать присутствие атрибута wsdl:required со значением "false" *для элемента-расширения в смысле необязательности этого элемента в сообщениях, генерируемых по WSDL-описанию.*

5.2 Транспорт и сообщения

5.2.1 Рекомендации по расширению структуры SOAP-сообщений

В разделе «4.2 Транспорт и сообщения» было показано, что в качестве протокола следует применять SOAP, при этом:

- реализация должна соответствовать WS-I Basic Profile;
- в качестве транспортных протоколов следует использовать HTTP(S) и SMTP;
- желательно использование механизмов WS-ReliableMessaging.

Для поддержки транзакциионности в среде Web-сервисов в реализацию протокола SOAP в рамках СУВ должны быть включены расширения для передачи в заголовках SOAP-сообщений координирующего контекста, описанного в разделе «5.6.1 Описание расширения структуры стандартных SOAP-сообщений».

Для поддержки безопасного обмена сообщениями в среде Web-сервисов в реализацию протокола SOAP в рамках СУВ должны быть включены расширения, позволяющие использовать привязку SOAP-сообщений к протоколу HTTPS, выбор которого обоснован в разделе «5.5 Безопасность».

В качестве основных транспортных протоколов для передачи SOAP-сообщений в рамках СУВ рекомендуется использовать протоколы HTTP/HTTPS или SMTP. Обоснование выбора данных протоколов приведено в разделе «4.2.2.1 Транспортный уровень - HTTP, HTTP/S, SMTP». Описание способов привязки SOAP-сообщений к выбранным транспортным протоколам приведено в разделе «5.2.2 Описание способов инкапсуляции SOAP-сообщения в различные транспортные протоколы».

5.2.2Описание способов инкапсуляции SOAP-сообщения в различные транспортные протоколы

5.2.2.1Инкапсуляция в протокол HTTP

HTTP имеет хорошо известную модель взаимодействия и шаблон обмена сообщениями. Клиент идентифицирует сервер по URI, подсоединяется к нему с помощью TCP/IP сети, отсылает HTTP-сообщение-запрос и получает HTTP-сообщение-отклик по тому же TCP-соединению. HTTP полностью коррелирует сообщение-запрос и соответствующий ему сообщение-отклик, поэтому приложение, использующее эту привязку, может реализовать корреляцию между отправленным в теле HTTP-запроса SOAP-сообщением и SOAP-сообщением, возвращенным в качестве HTTP-отклика. Подобным образом, HTTP идентифицирует конечный сервер по URI, который может также служить идентификатором SOAP-узла сервера.

HTTP могут использовать многочисленные посредники между начальным клиентом и сервером, инициировавшим обмен сообщениями, идентифицируемые по URI-запросу, в этом случае модель запрос/отклик является по существу сериями таких пар.

HTTP-привязка использует функцию SOAP Web-метода, позволяющую приложениям выбирать один из так называемых Web-методов – GET или POST – чтобы использовать для обмена сообщениями с помощью HTTP. Кроме того, она использует два шаблона обмена сообщениями, которые дают приложениям два способа обмена SOAP-сообщениями посредством HTTP:

- использование HTTP-метода POST для передачи SOAP-сообщений в теле HTTP-запроса и HTTP-отклика;
- использование HTTP-метода GET в HTTP-запросе для возвращения SOAP-сообщения в теле HTTP-отклика.

Первый шаблон использования является HTTP-реализацией функции привязки - шаблона обмена SOAP-сообщениями типа "запрос-отклик", второй – шаблона обмена SOAP-сообщениями типа "отклик".

Цель разработки этих двух способов - реализовать две парадигмы взаимодействия, одинаково хорошо подходящие для World Wide Web. Первый тип взаимодействия позволяет использовать данные в теле HTTP-метода POST для создания или изменения состояния ресурса, идентифицируемого по URI, в соответствии с которым направлен HTTP-запрос. Второй тип шаблона взаимодействия дает возможность использовать HTTP-запрос GET для получения представления о ресурсе без какого-либо изменения его состояния. В первом случае касающийся SOAP аспект вопроса состоит в том, что тело HTTP-запроса POST является SOAP-сообщением, которое, кроме того, что должно быть обработано (согласно модели обработки SOAP) в соответствии со специфичной для приложения обработкой, должно также соответствовать семантике POST. Во втором случае типичной реализацией является получение представления запрашиваемого ресурса в виде SOAP-сообщения, а не в виде HTML- или XML-документа. То есть HTTP-заголовок типа содержимого сообщения идентифицирует это как медиа-тип "application/soap+xml".

Следующим аспектом HTTP-привязки SOAP является проблема определения, какой из вышеуказанных двух шаблонов обмена сообщениями использовать. Спецификация SOAP содержит руководство, описывающее, в каких обстоятельствах приложения могут использовать тот или иной определенный шаблон обмена сообщениями. Шаблон обмена SOAP-сообщениями с использованием HTTP-метода GET используется в том случае, если приложение использует этот шаблон только для поиска информации, а сам ресурс в результате взаимодействия остается "нетронутым". Подобные взаимодействия трактуются в спецификации HTTP как безопасные и идемпотентные. Поскольку использование SOAP HTTP-метода GET не разрешено для SOAP-сообщения, содержащегося в запросе, приложения, которым необходим доступ к функциям взаимодействия с внешними объектами, и которые поддерживаются только специфичным для привязки выражением внутри информационного множества SOAP (то есть в качестве заголовочных SOAP-блоков), не могут использовать этот шаблон обмена сообщениями. Необходимо отметить, что HTTP-метод POST привязки может применяться во всех случаях.

Следующие подразделы демонстрируют примеры использования этих двух шаблонов обмена сообщениями, определенных для HTTP-привязки.

5.2.2.1.1 Использование в SOAP HTTP-метода GET

Использование HTTP-привязки совместно с шаблоном обмена SOAP-сообщениями типа "отклик" ограничивается HTTP-методом GET. Это означает, что откликом на HTTP-запрос GET, запрашивающего с SOAP-узла, будет являться SOAP-сообщение в HTTP-отклике.

Пример 1 показывает HTTP-метод GET, направленный приложением путешественника по URI

```
http://travelcompany.example.org/reservations?id=F12G3H123,
```

где можно увидеть маршрут путешествия.

Пример 1

```
GET /travelcompany.example.org/reservations?id=F12G3H123  HTTP/1.1
Host: travelcompany.example.org
Accept: text/html;q=0.5, application/soap+xml
```

В этом примере медиа-тип "application/soap+xml" более предпочтителен для интерпретации компьютером-клиентом, нежели медиа-тип "text/html" - для интерпретации браузером-клиентом человека.

Пример 2 показывает HTTP-отклик на запрос GET. Тело HTTP-отклика содержит SOAP-сообщение, показывающее детали путешествия.

Пример 2

```
HTTP/1.1 200 OK
Content-Type: application/soap+xml; charset="utf-8"
Content-Length: nnnn

<?xml version='1.0' ?>
<env:Envelope xmlns:env="http://www.w3.org/2003/05/soap-envelope">
 <env:Header>
  <m:reservation xmlns:m="http://travelcompany.example.org/reservation"
       env:role="http://www.w3.org/2003/05/soap-envelope/role/next"
           env:mustUnderstand="true">
   <m:reference>uuid:093a2da1-q345-739r-ba5d-pqff98fe8j7d</m:reference>
   <m:dateAndTime>2001-11-30T16:25:00.000-05:00</m:dateAndTime>
  </m:reservation>
 </env:Header>
 <env:Body>
```

```
  <rdf:RDF xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
            xmlns:x="http://travelcompany.example.org/vocab#"
      env:encodingStyle="http://www.w3.org/1999/02/22-rdf-syntax-ns#">
    <x:ReservationRequest
 rdf:about="http://travelcompany.example.org/reservations?code=FT35ZBQ">
      <x:passenger>Eke Jygvan Шyvind</x:passenger>
      <x:outbound>
       <x:TravelRequest>
        <x:to>LAX</x:to>
        <x:from>LGA</x:from>
        <x:date>2001-12-14</x:date>
       </x:TravelRequest>
      </x:outbound>
      <x:return>
       <x:TravelRequest>
        <x:to>JFK</x:to>
        <x:from>LAX</x:from>
        <x:date>2001-12-20</x:date>
       </x:TravelRequest>
      </x:return>
    </x:ReservationRequest>
  </rdf:RDF>
 </env:Body>
</env:Envelope>
```

Необходимо отметить, что вышеприведенным примером показан случай, когда приложение возвращает состояние ресурса в ориентированном на представление данных виде (в виде SOAP-сообщения), который может быть обработан автоматически, в отличие от (X)HTML-документа, который может быть обработан браузером. Действительно, как правило, принимающим приложением является не браузер.

Также, как показано в примере, использование SOAP в теле HTTP-отклика дает возможность реализации специфичных для приложений функций, с помощью SOAP-заголовков. Посредством SOAP приложение получает практичную и непротиворечивую структуру, а также модель обработки для реализации таких функций.

5.2.2.1.2 Использование в SOAP HTTP-метода POST

Использование HTTP-привязки совместно с шаблоном обмена SOAP-сообщениями типа "запрос-отклик" ограничивается использованием HTTP-метода POST. Необходимо

отметить, что использование этого шаблона обмена сообщениями в HTTP-привязке доступно всем приложениям, используют ли они обмен общими XML-данными или обмен RPC, инкапсулированными в SOAP-сообщения (как в следующих примерах).

Примеры 4 и **5** показывают пример HTTP-привязки, использующей шаблон обмена сообщениями типа "запрос-отклик", а именно передачу RPC и ее возврат в теле SOAP-сообщения соответственно. Примеры и содержание этого раздела сконцентрированы исключительно на HTTP-заголовках и их роли.

Пример 4

```
POST /Reservations HTTP/1.1
Host: travelcompany.example.org
Content-Type: application/soap+xml; charset="utf-8"
Content-Length: nnnn

<?xml version='1.0' ?>
<env:Envelope xmlns:env="http://www.w3.org/2003/05/soap-envelope" >
 <env:Header>
   <t:transaction
           xmlns:t="http://thirdparty.example.org/transaction"
           env:encodingStyle="http://example.com/encoding"
           env:mustUnderstand="true" >5</t:transaction>
 </env:Header>
 <env:Body>
  <m:chargeReservation
     env:encodingStyle="http://www.w3.org/2003/05/soap-encoding"
          xmlns:m="http://travelcompany.example.org/">
   <m:reservation xmlns:m="http://travelcompany.example.org/reservation">
    <m:code>FT35ZBQ</m:code>
   </m:reservation>
    <o:creditCard xmlns:o="http://mycompany.example.com/financial">
     <n:name xmlns:n="http://mycompany.example.com/employees">
           Eke Jygvan Шyvind
     </n:name>
     <o:number>123456789099999</o:number>
     <o:expiration>2005-02</o:expiration>
    </o:creditCard>
   </m:chargeReservation>
  </env:Body>
```

```
</env:Envelope>
```

Пример 4 демонстрирует RPC-запрос, направленный приложению путешествий. SOAP-сообщение отправляется в теле HTTP-метода POST по URI, идентифицирующего ресурс "Reservations" на сервере travelcompany.example.org.

При использовании HTTP URI-запрос указывает на ресурс, которому был "отправлен" вызов. Кроме требования, что URI должен быть корректным, SOAP не накладывает больше никаких формальных ограничений на вид URI-запроса. Однако одним из архитектурных принципов Web является требование, чтобы все важные ресурсы были идентифицированы URI. Данный факт позволяет предположить, что правильно построенные с точки зрения архитектуры Web SOAP-сервисы будут представлять собой ряд ресурсов, каждый со своим собственным URI. Действительно, многие подобные ресурсы, скорее всего, будут создаваться динамически во время работы сервиса.

Если SOAP-сообщение помещается в тело HTTP-сообщения, необходимо выбрать "application/soap+xml" в качестве значения HTTP-заголовка Content-type.

Пример 5 показывает отклик RPC (его детали опущены для краткости), отправленный приложением бронирования путешествий в соответствующем HTTP-отклике на запрос **примера 4**. SOAP, используя HTTP-транспорт, полагается на семантику HTTP-кодов статуса для указания в HTTP информации о статусе. Например, серия кодов статуса 2xx показывает, что запрос клиента (включая SOAP-составляющую) был успешно получен, понят, акцептован и т. д.

Пример 5

```
HTTP/1.1 200 OK
Content-Type: application/soap+xml; charset="utf-8"
Content-Length: nnnn

<?xml version='1.0' ?>
<env:Envelope xmlns:env="http://www.w3.org/2003/05/soap-envelope" >
 <env:Header>
       ...
       ...
 </env:Header>
 <env:Body>
       ...
```

```
      ...
 </env:Body>
</env:Envelope>
```

При возникновении ошибки обработки запроса, спецификация HTTP-привязки требует использования HTTP 500 "Internal Server Error" с вложенным SOAP-сообщением, содержащим SOAP-ошибку, указывающую ошибку обработки на стороне сервера.

Пример 6 представляет собой сообщение об ошибке, однако на этот раз в него добавлены HTTP-заголовки.

Пример 6

```
HTTP/1.1 500 Internal Server Error
Content-Type: application/soap+xml; charset="utf-8"
Content-Length: nnnn

<?xml version='1.0' ?>
<env:Envelope xmlns:env="http://www.w3.org/2003/05/soap-envelope">
  <env:Body>
    <env:Fault>
     <env:Code>
       <env:Value>env:Sender</env:Value>
       <env:Subcode>
        <env:Value>rpc:BadArguments</env:Value>
       </env:Subcode>
     </env:Code>
     <env:Reason>
      <env:Text xml:lang="en-US">Processing error</env:Text>
      <env:Text xml:lang="cs">Chyba zpracovбnн</env:Text>
     </env:Reason>
     <env:Detail>
      <e:myFaultDetails
        xmlns:e="http://travelcompany.example.org/faults" >
        <e:message>Name does not match card number</e:message>
        <e:errorcode>999</e:errorcode>
      </e:myFaultDetails>
     </env:Detail>
   </env:Fault>
 </env:Body>
```

```
</env:Envelope>
```

5.2.2.2Инкапсуляция в протокол SMTP (X.400)

Разработчики приложений могут использовать также email-инфраструктуру для передачи SOAP-сообщений, причем как текст email-сообщений так и их вложения. Примеры, приведенные ниже, иллюстрируют такой метод передачи SOAP-сообщений, однако они не должны трактоваться как некие стандартные способы реализации этого метода. Спецификации SOAP Версия 1.2 не специфицируют подобную привязку, хотя существует *неофициальный* W3C Note, описывающий email-привязку для SOAP. Его основная цель - продемонстрировать применение общей Структуры Протокольной Привязки SOAP.

Пример 7 показывает сообщение, реализующее запрос на бронирование путешествия из **примера 4**, передаваемый как email-сообщение между отправляющим и принимающим mail-агентами пользователя. (Подразумевается, что отправляющий узел также умеет интерпретировать SOAP и способен обрабатывать любые SOAP-ошибки, полученные в ответ на исходящее сообщение, а также коррелировать любые входящие ответные SOAP-сообщения).

Пример 7

```
From: a.oyvind@mycompany.example.com
To: reservations@travelcompany.example.org
Subject: Travel to LA
Date: Thu, 29 Nov 2001 13:20:00 EST
Message-Id: <EE492E16A090090276D208424960C0C@mycompany.example.com>
Content-Type: application/soap+xml

<?xml version='1.0' ?>
<env:Envelope xmlns:env="http://www.w3.org/2003/05/soap-envelope">
 <env:Header>
  <m:reservation xmlns:m="http://travelcompany.example.org/reservation"
      env:role="http://www.w3.org/2003/05/soap-envelope/role/next"
         env:mustUnderstand="true">
   <m:reference>uuid:093a2da1-q345-739r-ba5d-pqff98fe8j7d</reference>
   <m:dateAndTime>2001-11-29T13:20:00.000-05:00</m:dateAndTime>
  </m:reservation>
  <n:passenger xmlns:n="http://mycompany.example.com/employees"
```

```
      env:role="http://www.w3.org/2003/05/soap-envelope/role/next"
         env:mustUnderstand="true">
   <n:name>Eke Jygvan Шyvind</n:name>
  </n:passenger>
 </env:Header>
 <env:Body>
  <p:itinerary
     xmlns:p="http://travelcompany.example.org/reservation/travel">
   <p:departure>
     <p:departing>New York</p:departing>
     <p:arriving>Los Angeles</p:arriving>
     <p:departureDate>2001-12-14</p:departureDate>
     <p:departureTime>late afternoon</p:departureTime>
     <p:seatPreference>aisle</p:seatPreference>
   </p:departure>
   <p:return>
     <p:departing>Los Angeles</p:departing>
     <p:arriving>New York</p:arriving>
     <p:departureDate>2001-12-20</p:departureDate>
     <p:departureTime>mid morning</p:departureTime>
     <p:seatPreference/>
   </p:return>
  </p:itinerary>
  <q:lodging
     xmlns:q="http://travelcompany.example.org/reservation/hotels">
   <q:preference>none</q:preference>
  </q:lodging>
 </env:Body>
</env:Envelope>
```

Хотя email характеризуется односторонним обменом сообщениями и не гарантирует их доставку, транспортные протоколы, подобные Simple Mail Transport Protocol (SMTP), все же предоставляют механизм извещения о доставке, который, в случае с SMTP, называется Delivery Status Notification (DSN) и Message Disposition Notification (MDN). Извещения имеют вид email-сообщений, отправленных на email-адрес, указанный в заголовке email-сообщения. Приложения, так же как и email-пользователи, могут использовать эти механизмы для получения статуса передачи email-сообщения, однако эти извещения, будучи доставлены, являются сообщениями SMTP-уровня. Разработчик

приложений должен хорошо понимать возможности и ограничения этих извещений о доставке, также как и риск того, что извещение может не быть сформировано в случае успешной доставки email-сообщения.

SMTP-сообщения статуса доставки являются иными, нежели сообщения, обрабатываемые на SOAP-уровне. SOAP-отклики на содержащиеся в email-сообщении SOAP-данные, получаемые в результате, будут возвращены с помощью email-сообщения, которое может содержать (а может и не содержать) SMTP-ссылку на оригинальное email-сообщение. Использование заголовка *In-reply-to:* может помочь в достижении корреляции на SMTP-уровне, но не обязательно позволит коррелировать сообщения на SOAP-уровне.

Пример 8 демонстрирует SOAP-сообщение (детали тела сообщения для краткости опущены), отправленное приложением продажи билетов приложению бронирования путешествия, и проясняющее некоторые детали бронирования, кроме уже переданных email-сообщением. В этом примере *Message-Id* оригинального email-сообщения передан в дополнительном email-заголовке *In-reply-to:*, который коррелирует email-сообщения на SMTP-уровне, однако не позволяет корреляцию на SOAP-уровне. В этом примере приложение для корреляции SOAP-сообщений использует заголовочный блок *reservation* **То**, каким образом эта корреляция может быть реализована, зависит от конкретного приложения и не является предметом рассмотрения SOAP.

Пример 15

```
From: reservations@travelcompany.example.org
To: a.oyvind@mycompany.example.com
Subject: Which NY airport?
Date: Thu, 29 Nov 2001 13:35:11 EST
Message-Id: <200109251753.NAA10655@travelcompany.example.org>
In-reply-to:<EE492E16A090090276D208424960C0C@mycompany.example.com>
Content-Type: application/soap+xml

<?xml version='1.0' ?>
<env:Envelope xmlns:env="http://www.w3.org/2003/05/soap-envelope">
 <env:Header>
  <m:reservation xmlns:m="http://travelcompany.example.org/reservation"
     env:role="http://www.w3.org/2003/05/soap-envelope/role/next"
       env:mustUnderstand="true">
   <m:reference>uuid:093a2da1-q345-739r-ba5d-pqff98fe8j7d</reference>
```

```
   <m:dateAndTime>2001-11-29T13:35:00.000-05:00</m:dateAndTime>
  </m:reservation>
  <n:passenger xmlns:n="http://mycompany.example.com/employees"
      env:role="http://www.w3.org/2003/05/soap-envelope/role/next"
        env:mustUnderstand="true">
   <n:name>Eke Jygvan Шyvind</n:name>
  </n:passenger>
 </env:Header>
 <env:Body>
  <p:itinerary
     xmlns:p="http://travelcompany.example.org/reservation/travel">
   <p:itineraryClarifications>
      ...
      ...
   </p:itineraryClarifications>
  </p:itinerary>
 </env:Body>
</env:Envelope>
```

5.3 Каноническая модель данных и язык спецификации схем данных

5.3.1 Руководство по процессу разработки XML-схем

Цель этого раздела – предоставить набор нормативов, помогающих в разработке XML-схем в соответствие со стандартами СЭВ и ИС ЭАР.

В первую очередь, определим роль моделирования, и в частности UML-моделирования, при разработке XML-схем. Язык XML Schema не богат семантически, и содержит массу технических деталей представления данных, что требует дополнения его концептуальными моделями этих данных.

Модели играют ведущую роль в разработке систем. Большие корпоративные системы часто сложны для понимания. В таких случаях порой необходимо разбить исходную проблему на множество вспомогательных подмоделей, представляющих различные точки зрения. В каждой из этих подмоделей целенаправленно игнорируются некоторые аспекты системы, которые не относятся к их области применения. Такая декомпозиция является фундаментальным способом справиться со сложностью реального мира, опуская ненужные детали, мешающие нам сконцентрироваться на самой

задаче. К тому же группам, работающим с системой на различных этапах, требуются модели с различной степенью абстрагирования.

В контексте системной интеграции уровня B2B, все бизнес-партнеры должны договориться об общей информационной модели, в рамках которой был бы определен словарь для коммуникации, ориентированный на конкретные задачи. Модели включают как структуру данных для XML-документов, которые участвуют в документообороте, так и процессы, моделирующие расширенные диалоги, требующиеся для выполнения сложных бизнес-транзакций.

При разработке XML-схем для обмена данными в ПЭВ или ЭАР, рекомендуется руководствоваться не техническими деталями представления данных в XML, а в первую очередь учитывать спецификации дизайна, модели данных систем. Рекомендуется, чтобы в процессе разработки XML-схем возникали следующие спецификации:

- Спецификации требований
- UML-диаграмма классов
- UML-диаграммы форматов сообщений в ПЭВ / ЭАР
- UML-диаграмма последовательности ПЭВ / ЭАР
- Наконец, XSD файл(ы), отражающие технические детали представления этой информации в XML-формате при обмене данными.

При разработке конкретно XSD-файлов следует придерживаться предписаний раздела «Руководство по стилю XML-схем».

Следующий подраздел описывает применение UML для описания концептуальной модели области приложений, и XML-схемы обмена данными между этими приложениями как частного случая.

5.3.1.1 UML-моделирование при разработке XML-схем

В данном разделе мы приведём пример использования UML-модели для описания XML-схемы. Мы будем работать со словарем для языка заказов, который определялся в разделе 2.1 документа XML Schema Part 0: Primer и далее используется во всей спецификации W3C.

Ниже приведена XSD-схема для этого примера (фрагмент XML Schema Part 0: Primer):

```
<schema targetNamespace="http://www.example.com/IPO"
        xmlns="http://www.w3.org/2001/XMLSchema"
        xmlns:ipo="http://www.example.com/IPO">
```

```
<annotation>
 <documentation xml:lang="en">
  International Purchase order schema for Example.com
  Copyright 2000 Example.com. All rights reserved.
 </documentation>
</annotation>

<!-- include address constructs -->
<include
 schemaLocation="http://www.example.com/schemas/address.xsd"/>

<element name="purchaseOrder" type="ipo:PurchaseOrderType"/>

<element name="comment" type="string"/>

<complexType name="PurchaseOrderType">
 <sequence>
  <element name="shipTo"     type="ipo:Address"/>
  <element name="billTo"     type="ipo:Address"/>
  <element ref="ipo:comment" minOccurs="0"/>
  <element name="items"      type="ipo:Items"/>
 </sequence>
 <attribute name="orderDate" type="date"/>
</complexType>

<complexType name="Items">
 <sequence>
  <element name="item" minOccurs="0" maxOccurs="unbounded">
   <complexType>
    <sequence>
     <element name="productName" type="string"/>
     <element name="quantity">
      <simpleType>
       <restriction base="positiveInteger">
        <maxExclusive value="100"/>
       </restriction>
      </simpleType>
     </element>
```

```
      <element name="USPrice"    type="decimal"/>
      <element ref="ipo:comment" minOccurs="0"/>
      <element name="shipDate"   type="date" minOccurs="0"/>
     </sequence>
     <attribute name="partNum" type="ipo:SKU" use="required"/>
    </complexType>
   </element>
  </sequence>
 </complexType>

 <simpleType name="SKU">
  <restriction base="string">
   <pattern value="\d{3}-[A-Z]{2}"/>
  </restriction>
 </simpleType>

</schema>
```

В нашей модели дополнительно определены международные адреса и поддержка нескольких схем, как описано в разделе 4.1 спецификации W3C. Ниже приведена эта схема (фрагмент XML Schema Part 0: Primer):

```
<schema targetNamespace="http://www.example.com/IPO"
        xmlns="http://www.w3.org/2001/XMLSchema"
        xmlns:ipo="http://www.example.com/IPO">

 <annotation>
  <documentation xml:lang="en">
   Addresses for International Purchase order schema
   Copyright 2000 Example.com. All rights reserved.
  </documentation>
 </annotation>

 <complexType name="Address">
  <sequence>
   <element name="name"   type="string"/>
   <element name="street" type="string"/>
   <element name="city"   type="string"/>
  </sequence>
 </complexType>
```

```
<complexType name="USAddress">
 <complexContent>
  <extension base="ipo:Address">
   <sequence>
    <element name="state" type="ipo:USState"/>
    <element name="zip"   type="positiveInteger"/>
   </sequence>
  </extension>
 </complexContent>
</complexType>

<complexType name="UKAddress">
 <complexContent>
  <extension base="ipo:Address">
   <sequence>
    <element name="postcode" type="ipo:UKPostcode"/>
   </sequence>
   <attribute name="exportCode" type="positiveInteger" fixed="1"/>
  </extension>
 </complexContent>
</complexType>

<!-- other Address derivations for more countries -->

<simpleType name="USState">
 <restriction base="string">
  <enumeration value="AK"/>
  <enumeration value="AL"/>
  <enumeration value="AR"/>
  <!-- and so on ... -->
 </restriction>
</simpleType>

<!-- simple type definition for UKPostcode -->

</schema>
```

Словарь для языка заказов определяется в двух модулях, соответствующих основному типу *PurchaseOrder* и отдельному модулю *Address*. В UML такие модули

называются пакетами. Спецификация первого пакета показана на диаграмме классов UML на иллюстрации 3. Класс *PurchaseOrder* имеет два атрибута и три ассоциации, которые определяют его структуру. Некоторые из этих атрибутов включают мощность, определенную символами [0..1], подразумевая, что значение этих атрибутов опционально и принимает значения либо 0, либо 1.

Класс *Address* играет роль как *shipTo*, так и *billTo* в ассоциации с *PurchaseOrder*. (Замечание: может статься, что *shipTo* и *billTo* являются дочерними элементами в схеме.) Мощность 1 обозначает, что для каждого *PurchaseOrder* должен существовать ровно 1 адрес. Заметим, что в классе *Item quantity* имеет тип *QuantityType*. Это тип определен как другой класс в UML-модели. На той же диаграмме *QuantityType* определен как подкласс класса *positiveInteger*, который объявляется как приходящий из пакета *XSD_Datatypes* в этой UML-модели. Таким образом, *quantity* является специальным типом положительных целых чисел.

И *QuantityType*, и *SKU* являются определенными пользователем типами данных, и оба включают атрибуты, что ограничивает их область использования. Атрибуты *pattern* и *maxExclusive* являются заданными значениями и используются на следующих этапах процесса проектирования для управления созданием XML Schema. Наконец, имя класса *Address* показывается курсивом; это означает, что это абстрактный класс, и что он не предназначен для непосредственного использования. Как далее будет видно, *Address* кроме того определяется в других диаграммах классов UML.

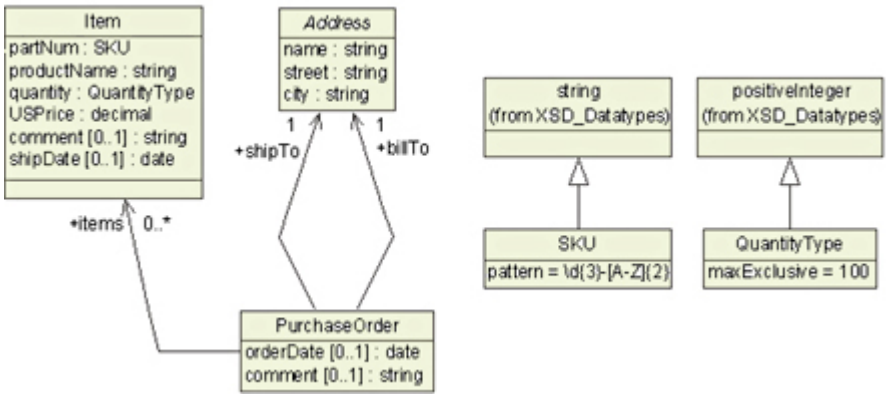


Рисунок 17. Концептуальная модель словаря заказов покупок

Спецификация пакета *Address*, изображенная на иллюстрации ниже, следует подобной логике. На этой диаграмме *USAddress* и *UKAddress* являются подтипами *Address*. В терминах объектно-ориентированного подхода это означает, что оба этих подтипа наследуют три атрибута, определенные в их суперклассе. Атрибут *exportCode* класса *UKAddress* инициализируется со значением 1.

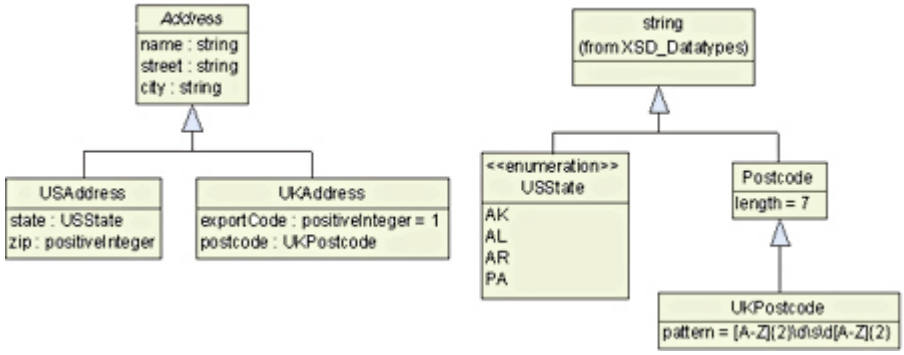


Рисунок 18. Компонент схемы Modularized Address

В качестве более сложного примера приведём также фрагмент UML-диаграммы схемы UDDI v3.0:

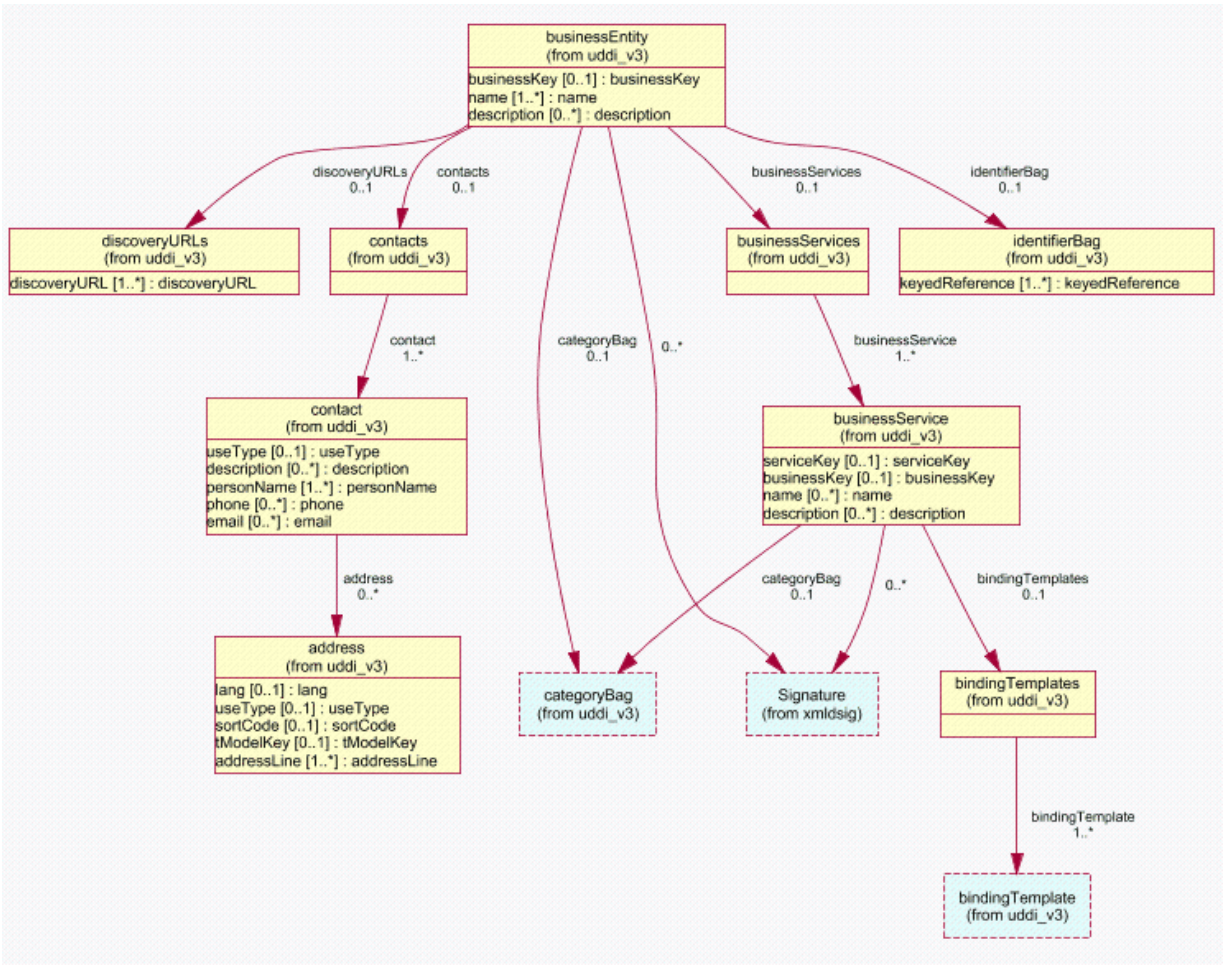


Рисунок 19. Пример UML-диаграммы схемы UDDI v3.0

5.3.1.2Инструментарий для преобразования между UML и XSD

В качестве инструментария для совместной разработки UML и XML-схем может быть использовано средство hyperModel (xmlmodeling.com). HyperModel представляет собой plugin к Eclipse Workbench, обладающий следующими возможностями:

- **Импорт любой XML-схемы в UML**
    - *hyper*Model создаёт понятные концептуальные модели для определений XML-схем.
- **Генерация XML-схемы по любой UML-модели**
    - *hyper*Model поддерживает полное отображение из языка моделирования UML в язык описания XML-схем.
- **Настройка дизайна генерируемой XML-схемы**
    - Для настройки дизайна схем используется UML-профиль, позволяющий аннотировать UML-модели настройками отображения в XSD.
- **Дополняет другие UML-инструментарии**
    - *hyper*Model дополняет другие UML-инструментарии за счёт обмена UML-моделями в формате XMI 1.0 [18].
    - Может быть использован совместно с Rational Rose®, Gentleware Poseidon®, ArgoUML и др.
- **Дополняет другие инструментарии для XML-дизайна**
    - Может использоваться для получения UML-диаграмм по XML-схеме, полученной другим инструментарием.
    - Инструментарии для дизайна XML-схем делают упор на иерархическую структуру документа, но уделяют мало внимания пониманию определений типов и ассоциаций в XML-словаре.
    - Определение XML-схемы в инструментарии вроде XML Spy® требует от разработчика полного понимания терминологии XML Schema, но эти детали мешают анализу и дизайну информационного наполнения.

5.3.2Руководство по стилю XML-схем

5.3.2.1Введение

Стандарты указывают XML как первичное средство для интеграции данных. Спецификации схем XML-документов должны соответствовать языку XML Schema, рекомендованному W3C.

Эта W3C-рекомендация допускает много вариантов дизайна XML-схем. Данный раздел предоставляет рекомендации и руководства в разработке XML-схем в рамках приложений и систем, совместимых со стандартами СЭВ и ИС ЭАР.

5.3.2.1.1 Подходы к дизайну XML-схем

Рассмотрим некоторые частные случаи разнообразия способов определения схем.

XML Schema допускает множество стилей определения и объявления, от полностью иерархического до полностью планарного стиля. Нет абсолютно верного или неверного стиля, в разных ситуациях следует применять различные стили (см. например, раздел «Глобальные определения»). В XML-сообществе некоторые стили получили собственные имена, рассмотрим их подробнее.

Одним вариантом крайности является модель «матрёшки», когда каждый элемент определяется локально в контексте использования. В таком стиле можно описать элемент «ФИО» (Name) следующим образом:

```
<?xml version="1.0" encoding="UTF-8"?>
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema"
elementFormDefault="qualified" attributeFormDefault="unqualified">
  <xs:element name="Name">
    <xs:complexType>
      <xs:sequence>
        <xs:element name="Forename" maxOccurs="unbounded">
          <xs:simpleType>
            <xs:restriction base="xs:string">
              <xs:minLength value="1"/>
              <xs:maxLength value="35"/>
            </xs:restriction>
          </xs:simpleType>
        </xs:element>
        <xs:element name="Surname">
          <xs:simpleType>
```

```
            <xs:restriction base="xs:string">
              <xs:minLength value="1"/>
              <xs:maxLength value="35"/>
            </xs:restriction>
          </xs:simpleType>
        </xs:element>
      </xs:sequence>
    </xs:complexType>
  </xs:element>
</xs:schema>
```

Здесь элемент *Name* определяет два подэлемента *Forename* и *Surname* внутри собственного контекста. Эти определения содержат вложенные анонимные определения простых типов данных.

Другая крайность – это иерархический подход, известный как «модель ломтиков салями». В этом подходе каждый элемент определяется глобально, что соответствует разбиению документа-экземпляра на отдельные компоненты:

```
<?xml version="1.0" encoding="UTF-8"?>
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema"
elementFormDefault="qualified" attributeFormDefault="unqualified">
  <xs:element name="Name">
    <xs:complexType>
      <xs:sequence>
        <xs:element ref="Forename" maxOccurs="unbounded"/>
        <xs:element ref="Surname"/>
      </xs:sequence>
    </xs:complexType>
  </xs:element>
  <xs:element name="Forename">
    <xs:simpleType>
      <xs:restriction base="xs:string">
        <xs:minLength value="1"/>
        <xs:maxLength value="35"/>
      </xs:restriction>
    </xs:simpleType>
  </xs:element>
  <xs:element name="Surname">
```

```
    <xs:simpleType>
      <xs:restriction base="xs:string">
        <xs:minLength value="1"/>
        <xs:maxLength value="35"/>
      </xs:restriction>
    </xs:simpleType>
  </xs:element>
</xs:schema>
```

Третий подход известен как «модель жалюзи». Он аналогичен подходу с «ломтиками», но вместо определения глобальных элементов, глобально объявляются только типы:

```
<?xml version="1.0" encoding="UTF-8"?>
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema"
elementFormDefault="qualified" attributeFormDefault="unqualified">
  <xs:element name="Name">
    <xs:complexType>
      <xs:sequence>
        <xs:element name="Forename" type="ForenameType"
                    maxOccurs="unbounded"/>
        <xs:element name="Surname" type="SurnameType"/>
      </xs:sequence>
    </xs:complexType>
  </xs:element>

  <xs:simpleType name="ForenameType">
    <xs:restriction base="xs:string">
      <xs:minLength value="1"/>
      <xs:maxLength value="35"/>
    </xs:restriction>
  </xs:simpleType>

  <xs:simpleType name="SurnameType">
    <xs:restriction base="xs:string">
      <xs:minLength value="1"/>
      <xs:maxLength value="35"/>
    </xs:restriction>
```

```
  </xs:simpleType>
</xs:schema>
```

Можно пойти дальше и совместить подходы «ломтиков» и «жалюзи», объявляя глобально и элементы, и типы:

```
<?xml version="1.0" encoding="UTF-8"?>
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema"
elementFormDefault="qualified" attributeFormDefault="unqualified">
  <xs:element name="Name">
    <xs:complexType>
      <xs:sequence>
        <xs:element ref="Forename" maxOccurs="unbounded"/>
        <xs:element ref="Surname"/>
      </xs:sequence>
    </xs:complexType>
  </xs:element>

  <xs:element name="Forename" type="ForenameType"/>
  <xs:element name="Surname" type="SurnameType"/>

  <xs:simpleType name="ForenameType">
    <xs:restriction base="xs:string">
      <xs:minLength value="1"/>
      <xs:maxLength value="35"/>
    </xs:restriction>
  </xs:simpleType>
  <xs:simpleType name="SurnameType">
    <xs:restriction base="xs:string">
      <xs:minLength value="1"/>
      <xs:maxLength value="35"/>
    </xs:restriction>
  </xs:simpleType>
</xs:schema>
```

В общем случае, в конкретной XML-схеме совмещают элементы разных стилей. Всякий документ, удовлетворяющей модели «матрёшки» или «жалюзи» будет удовлетворять остальным стилям. Однако модель «ломтиков» и совмещённая модель

допускают использование *Forename* и *Surname* в качестве корневого элемента в документе, а модель «матрёшки» и «жалюзи» - позволяют только элементу *Name* быть корневым.

5.3.2.1.2 Секции и терминология

Каждое указание разбито на две или три секции:

- "Рекомендация" кратко излагает требование или рекомендацию
- "Пояснение" описывает причины такого выбора и поясняет детали
- "Примеры" могут указывать ненормативные примеры соблюдения данного руководства

5.3.2.2Общие указания по XML-схемам

5.3.2.2.1 Первичный язык схем

5.3.2.2.1.1 Рекомендация

В качестве основного языка описания схем XML-документов должен быть использован язык W3C XML Schema.

5.3.2.2.1.2 Пояснение

В целях интероперабельности важно, чтобы все системы использовали один и тот же язык для описания схем, иначе определения общих схем нельзя будет использовать повторно. В настоящее время, рекомендуется к использованию язык XML Schema.

В некоторых случаях есть необходимость указать дополнительные ограничения, помимо тех, которые могут быть указаны W3C XML Schema. В таком случае можно использовать совместимые расширения Schematron [44] для проверки специфичных для приложения ограничений.

5.3.2.2.2 Сложность схем

5.3.2.2.2.1 Рекомендация

Следует помнить о целевой аудитории пользователей схемы. Мало употребимые конструкции языка XML Schema не желательно использовать, когда есть более простые альтернативы. Разработчикам схем желательно обращаться к примерам других схем, описанных в стандартах СЭВ и ИС ЭАР, чтобы определить желательный стиль схем. Разработчикам схем желательно принимать во внимание удобство тестирования их схем.

5.3.2.2.2 Пояснение

Язык XML Schema крайне гибок и выразителен. Во многих случаях схемы, следующие одной и той же цели, могут быть сделаны очень простыми или очень сложными. Многие люди, которым придётся общаться со схемами, могут слабо владеть XML Schema, поэтому не следует делать их чрезмерно сложными.

Средства разработки схем и инструменты тестирования содержат ошибки, в основном в поддержке редко используемых аспектов языка. Простые схемы не только проще тестировать, но и больше уверенность в стабильной работе инструментария.

5.3.2.2.3 Моделировать данные, а не бланки

5.3.2.2.3.1 Рекомендация

Желательно, чтобы XML-схемы моделировали данные, необходимые для приложений, а не существующие бланки/формы или существующие форматы сообщений. Тогда как существующие бланки/формы и форматы сообщений часто – отличная отправная точка в разработке схемы, нежелательно, чтобы они определяли конечный дизайн сообщения.

5.3.2.2.3.2 Пояснение

Это требование обусловлено двумя причинами: во-первых, хорошо спроектированный бланк спроектирован для его представления на бумаге, а не на экране компьютера, и существующий формат сообщений не всегда точно отражает информацию. Во-вторых, дизайн схемы должен следовать из информационной модели и моделей взаимодействия ПЭВ/ЭАР, который должен быть спроектирован с нуля, а не слепо следуя сложившимся процессам и существующим системам.

5.3.2.2.4 Использование «смешанной модели содержимого»: `mixed="true"`

5.3.2.2.4.1 Рекомендация

Желательно, чтобы XML-схемы моделировали «дата-центричные» документы, ориентированные на представление XML-документом объектных данных. Такой XML содержит элементы, которые в свою очередь содержат подэлементы, и только самые глубокие подэлементы – содержат символьные данные, то есть, не допускается «смешанной модели содержимого», когда *`mixed="true"`*.

Альтернативой «дата-центричным» являются «текст-центричные» документы, такие как XHTML, ориентированные на представление слабоструктурированного текста.

Как правило, весь документ целиком является либо «дата-центричным», либо «текст-центричным». Тем не менее, это не всегда так, например, ориентированный на данные документ может содержать XHTML-документацию.

Желательно, чтобы «дата-центричные» элементы не допускали «смешанной модели содержимого». Атрибут *mixed="true"* может быть использован только для элементов, содержащих слабоструктурированную документацию.

5.3.2.2.4.2 Пояснение

XML-схемы в СЭВ и ИС ЭАР используются для описания структур данных участвующих в средах информационных источников и Web-сервисов, а не для описания документов, ориентированных на текст, таких как XHTML. Соответственно, основой структуры документов должен быть дата-центричный подход, а текст-центричные вставки должны допускаться только в виде альтернативы строкам (xsd:string), для указания слабоструктурированного текста, когда планарного текста недостаточно.

5.3.2.2.4.3 Примеры

Пример ориентированного на текст XML-формата:

```
<Paragraph>This is an example of a <Emphasize>mixed</Emphasize> content model in a
text-centric document, and is acceptable.</Paragraph>
```

Пример ориентированного на данные XML-формата:

```
<vCard:N>
  <vCard:Given> Иван </vCard:Given>
  <vCard:Other> Иванович </vCard:Other>
  <vCard:Family> Иванов </vCard:Family>
</vCard:N>
```

5.3.2.2.5 Использование пространств имён и префиксов

5.3.2.2.5.1 Рекомендация

Если схема имеет целевое пространство имён, то в документе со схемой должно использоваться пространство имён по умолчанию (default namespace), и оно должно совпадать с целевым пространством имён.

Пространство имён W3C XML Schema должно быть квалифицировано префиксом *xsd* или *xs*. Подходящий префикс должен быть использован и для всех остальных пространств имён.

5.3.2.2.5.2 Пояснение

Как правило, определение пространства имён по умолчанию таким же, как целевое пространство имён – разумный шаг. Любой другой подход может привести к проблемам, если другой документ со схемой без целевого пространства имён будет включён (*<include>)* в разрабатываемую схему.

Поскольку это означает, что ни пространство имён XML-схемы, ни любое другое пространство имён не может быть пространством имён по умолчанию, то им нужно указать префикс. Все инструментарии разработки схем используют для этого префикс *xs* или *xsd*, поэтому предлагается использовать один из этих вариантов.

Это делает использование пространств имён явным и предоставляет разработчикам схем большую гибкость в их использовании.

5.3.2.2.6 Несколько документов XML-схем в одном пространстве имён

5.3.2.2.6.1 Рекомендация

В случае импорта нескольких схем, определённых в одном и том же целевом пространстве имён, в некоторую другую схему, следует определить документ, который включает эти документы в себя с помощью *xs:include*, а затем использовать *xs:import* для импорта этого документа.

5.3.2.2.6.2 Пояснение

Рекомендация XML Schema не уточняет способ интерпретации нескольких схем, имеющих одно и то же целевое пространство имён. В результате, если такие документы схем импортируются в другую схему с помощью *xs:import*, разные реализации XML-процессоров поступают по-разному. Некоторые позволяют это сделать, некоторые считают это ошибкой, а некоторые просто обрабатывают только первый *xs:import*. Проще вместо использования нескольких *xs:import* создать документ, который включает эти документы в себя с помощью *xs:include*, а затем использовать *xs:import* для импорта этого документа.

5.3.2.2.7 Использование `elementFormDefault`

5.3.2.2.7.1 Рекомендация

elementFormDefault должен иметь значение *qualified*.

5.3.2.2.7.2 Пояснение

Это гарантирует, что разработчик, читающий XML-экземпляр или использующий схему повторно, может опираться на видимые префиксы для определения того, какие элементы куда относятся, а не изучать для этого детально структуру схем.

5.3.2.2.8 Версии схем

5.3.2.2.8.1 Рекомендация

В соответствие с текущей практикой W3C, схемам желательно указывать номер версии схемы в атрибуте *version* элемента *schema*. Считается, что этот номер версии относится ко всем компонентам, определённым в схеме.

В случае, если этого недостаточно, номер версии может также быть указан в виде даты на конце целевого пространства имён, чтобы указать год и месяц (и опционально - день) публикации (например, http://sample.ru/schemas/someschema-200402).

5.3.2.2.8.2 Пояснение

Указание номера версии схемы помогает избежать проблем и путаницы при ошибочном использовании неверной версии схемы.

Во многих случаях достаточно указать номер версии с помощью атрибута *version*. Тем не менее, в некоторых случаях документу-экземпляру схемы сложно указать, какой версии схемы он подчиняется. Например, если для некоторой общеприменимой схемы документ не использует *schemaLocation* для указания положения схемы. В этих случаях можно использовать пространство имён с номером версии для чёткости спецификации.

5.3.2.2.9 Именование файлов схем

5.3.2.2.9.1 Рекомендация

Желательно, чтобы изданные документы со схемами указывали полный номер версии схемы в имени файла в формате *<имя файла>-vn-m.xsd*, где *n* и *m* – главный и второстепенный номер версии, а *v* – это символ "v".

Если схема публикуется с постоянной периодичностью (годовой и пр.), то имя файла может содержать код даты. Если дата указывается, то желательно указывать её как в приведённом ниже примере, в формате *ccyy* (4-симовльный год) или *ccyymm* (год и месяц).

5.3.2.2.9.2 Пояснение

При соблюдении данного правила гарантируется целостность приложений, поскольку никакая схема, ссылающаяся на другую схему с помощью директив *include*

или *import,* не станет ссылаться на изменённую версию этой схемы без явного изменения в файле, содержащем ссылку.

5.3.2.2.9.3 Пример

Для схемы с годичным циклом выпуска: *myschema-2003-v1-2.xsd*

5.3.2.2.10 Относительные и абсолютные ссылки

5.3.2.2.10.1 Рекомендация

В случае, когда схемы тесно связаны (и соответственно вероятно хранятся в файловой системе в одинаковом расположении друг относительно друга) в include и import-выражениях, желательно использовать относительные ссылки. В случае, когда связь схем слабая (когда они в основе своей независимы), желательно использовать абсолютные ссылки.

5.3.2.2.10.2 Пояснение

Документы со схемами могут быть перемещены из одного места хранения в другое. В таких случаях, должна быть поддержана целостность ссылок в схемах. В случае, когда вероятно, что схемы будут перемещены совместно как одна группа, использование относительных ссылок между ними обеспечит целостность ссылок. Напротив, документы схем, которые с большой вероятностью будут перемещены отдельно, а не в составе группы, или вынесены из группы – должны использовать абсолютные ссылки на другие схемы.

5.3.2.2.10.3 Примеры

Ниже показан пример, содержащий *include* схемы *another-schema.xsd* и *import* схемы schema2*.xsd*. Первая указывается абсолютным URL, вторая имеет общую тематику с импортирующей схемой и хранится вместе с остальными такими схемами, поэтому для ссылки на неё указывается относительный URL.

```
<xs:schema
  targetNamespace="http://sample.ru/schema1"
  xmlns=" http://sample.ru/schema1"
  xmlns:xs="http://www.w3.org/2001/XMLSchema"
  elementFormDefault="qualified"
  version="1.0">

  <xsd:include
    schemaLocation="http://another-sample.ru/another-schema.xsd"/>
```

```
<xsd:import
  namespace="http://sample.ru/schema1"
  schemaLocation="schema2.xsd"/>
...
```

5.3.2.2.11 Примеры документов, удовлетворяющих схеме

5.3.2.2.11.1 Рекомендация

С каждой опубликованной схемой желательно указать хотя бы один пример правильного XML-документа, подчиняющегося этой схеме.

5.3.2.2.11.2 Пояснение

Указывать примеры документов-экземпляров схемы полезно по нескольким причинам. Во-первых, они помогают понять схему. Во-вторых, генерация экземпляра документа по схеме может помочь выявить недостатки схемы, которые усложняют генерацию экземпляра, или делают его сложно читаемым или обрабатываемым. Наконец, многие инструментарии разработки XML-схем тщательную проверку схемы на правильность производят, только когда она используется для валидации какого-то документа-экземпляра, а не в процессе написания самой схемы как документа.

5.3.2.2.12 Определение типов против определения элементов

5.3.2.2.12.1 Рекомендация

Во многих случаях есть выбор между определением повторно-используемого компонента как XSD-типа (xsd:simpleType, xsd:complexType) или как элемента (xsd:element).

Компонент должен быть определён как тип, если:

- в разных контекстах он может использоваться с разными названиями элементов, например, в разных ролях, или
- ожидается, что на основе этого типа могут строиться другие типы (с помощью xsd:extension, xsd:restriction).

Компонент желательно определить как элемент, если:

- нет намерения строит на его основе другие компоненты (с помощью xsd:extension, xsd:restriction)
- элемент планируется использовать под одним и тем же именем.

5.3.2.2.12.2 Пояснение

Во многих случаях элемент следует использовать под одним и тем же именем. Например, если ИНН всегда указывается как элемент `inn`, то семантика этого термина будет известна, и две системы, использующие один и тот же элемент для указания ИНН, будут понимать друг друга. Следовательно, возможно построить словарь элементов с определённой всем известной семантикой и таким образом гарантировать интероперабельность систем.

В других случаях имя элемента может зависеть от контекста использования, хотя его содержимое, значение – иметь единый смысл. Например, адрес может иметь разный смысл и использоваться под разными названиями, такими как «адрес для писем», «домашний адрес», «адрес места работы». Следовательно, адрес должен быть определён как глобальный тип данных, который может быть использован как тип разных элементов.

Другой параметр выбора между определением элемента или типа данных – это наличие намерения создавать на его основе другие компоненты схемы. Используя тип данных, мы можем воспользоваться механизмом наследования, гарантировать понятность схемы, избежать использования `xs:redefine` для этих целей.

Во многих случаях имеет смысл определить и тип данных, и элемент. Тогда элемент можно будет использовать повторно с его зафиксированной семантикой, а на основе типа строить подходящие модифицированные подтипы.

Следует также учитывать тот факт, что глобально определённый элемент может быть использован как корневой элемент документа-экземпляра схемы.

5.3.2.2.13 Глобальные определения

5.3.2.2.13.1 Рекомендация

Желательно, чтобы документы схем определяли как глобальные только те определения компонентов (элементов, типов), которые:

- повторно используются в схеме, или
- должны быть доступны для повторного использования в других схемах, или
- планируется использовать их как корневые document-элементы в XML-данных.

5.3.2.2.13.2 Пояснение

Основная причина такого подхода – смягчение последствий изменений. Локальное определение компонентов схемы позволяет легко контролировать, кто ещё использует эти определения и что произойдёт, если это определение изменить.

5.3.2.2.14 Общие определения и пространства имён

5.3.2.2.14.1 Рекомендация

Специфичный для некоторой предметной области, тематики, специализации набор определений следует определять в собственном целевом пространстве имён. Тогда получившуюся общую схему можно будет импортировать в другие схемы с помощью `xs:import`. Во всех ссылках на эту схему будет указываться пространство имён этой общей схемы. Желательно, чтобы общие схемы, поддерживаемые в контексте различных проектов, бизнес-процессов имели различные собственные целевые пространства имён.

5.3.2.2.15 Элементы против атрибутов

5.3.2.2.15.1 Рекомендация

Схемы должны быть спроектированы так, чтобы XML-элементы являлись основными носителями информации в XML-документах. Атрибуты целесообразно использовать для представления простых вспомогательных метаданных о содержимом элемента.

Атрибуты не должны использоваться для уточнения других атрибутов, если это может привести к неоднозначности.

5.3.2.2.15.2 Пояснение

В отличие от элементов, атрибуты не могут содержать структурированные данные. В связи с этим предпочтительно использовать элементы как основные носители информационного содержания. Тем не менее, использование атрибутов для указания второстепенных метаданных о содержимом элемента (например, формата даты, единицы измерения и пр.) может упростить вид документов-экземпляров и сделать их более читабельными.

5.3.2.2.16 Представление альтернативных условий

5.3.2.2.16.1 Рекомендация

Альтернативные условия желательно представлять значением элемента или атрибута, но не отсутствием или присутствием элемента.

5.3.2.2.16.2 Пояснение

Некоторые разработчики используют присутствие пустого элемента для обозначения значения «да» и отсутствие его для обозначения «нет». Другие предпочитают, чтобы элемент всегда присутствовал, и его содержимое или атрибут содержали значение «да/нет». Это упрощает понимание данных в документе-экземпляре человеком, и поэтому рекомендуется придерживаться этого стиля.

5.3.2.2.17 Документирование схем

5.3.2.2.17.1 Рекомендация

При документировании схемы должно использовать элемент *documentation*, а не XML-комментарии.

5.3.2.2.17.2 Пояснение

Подэлемент *documentation* элемента *annotation* предназначен для документирования схем. Преимущество этого элемента перед XML-комментариями в том, что он является частью информации о схеме, используется, например, для подготовки пользовательской документации.

5.3.2.2.18 Применение механизмов повторного использования схем

5.3.2.2.18.1 Рекомендация

Желательно избегать использования *xs:redefine*.

5.3.2.2.18.2 Пояснение

Это позволяет избежать побочных эффектов в повторно используемых компонентах и повысить ясность и читабельность документов схем.

5.3.2.2.18.3 Рекомендация

xs:import нельзя использовать без указания атрибута *namespace*.

5.3.2.2.18.4 Пояснение

Это допускает безпрефиксные ссылки на компоненты внешнего пространство имён, что приводит к схемам, которые сложно тестировать и модифицировать, и для которых не видны зависимости повторного использования.

5.3.2.3Указания по компонентной модели XML-схем

5.3.2.3.1 Конвенции именования

5.3.2.3.1.1 Рекомендация

Желательно придерживаться «upper camel case»-стиля в именовании типов, «upper camel case» или «lower camel case» в именовании элементов и «lower camel case» в именовании атрибутов. Элемент желательно представлять «lower camel case», если он имеет смысл свойства, характеристики описания содержащей его сущности (элемента), и «upper camel case», если он имеет смысл названия типа сущности, описываемой этим элементом.

Под «camel case» подразумевается конвенция именования, в которой несколько слов сливаются в один идентификатор так, что каждое следующее слово начинается с большой буквы. В «upper camel case» с большой буквы начинается также и первое слово, в «lower camel case» первое слово начинается с маленькой буквы.

5.3.2.3.1.2 Пояснение

Цель этой рекомендации – выработать правило корректного использования «upper case» или «lower case» стилей в именовании элементов и атрибутов.

5.3.2.3.1.3 Рекомендация

Желательно не использовать сокращения. Чересчур длинных названий желательно избегать и выбирать лаконичные и информативные наименования. Общеизвестные сокращения, в том числе сокращения из одних заглавных букв, могут быть использованы. Тем не менее, следует учитывать, что сокращение, хорошо известное одному сообществу, может быть непонятно другим.

5.3.2.3.1.4 Пояснение

Цель этой рекомендации – сделать названия понятными в разных организациях и сделать схемы более понятными.

5.3.2.3.2 Использование наследования (`extension` и `restriction`)

5.3.2.3.2.1 Рекомендация

Если существующее определение не полностью удовлетворяет требованиям схемы, схема может использовать механизм наследования, чтобы определить новый тип на основе существующего.

В случае ограничения (xsd:restriction) простого типа (xsd:simpleType), представленного перечислением (xsd:enumeration) или шаблоном (xsd:pattern), следует убедиться, что ограничивающее определение удовлетворяет унаследованному, то есть сужает диапазон значений.

5.3.2.3.2.2 Пояснение

Механизм наследования позволяет сделать получение новых типов данных более понятным для пользователей схем, и позволяет инструментариям определять зависимости между определениями. Существует четыре типа наследования с помощью директив *`extension`* (расширение) и *`restriction (ограничение):`*

- ограничение простого типа данных
- расширение простого типа данных до сложного типа данных
- ограничение сложного типа данных

- расширение сложного типа данных.

Следует заметить, что при ограничении сложного типа данных синтаксис XML Schema требует почти полного повторения определения базового типа. Это может существенно усложнить идентификацию необходимых изменений в производных типах при изменении определения базового типа. В связи с этим следует быть предельно аккуратным при ограничении сложных типов данных.

5.3.2.3.3 Использование атрибутов `default` *и* `fixed`

5.3.2.3.3.1 Рекомендация

Атрибут *`default`* нежелательно использовать для добавления важной информации в атрибуты или элементы в XML-документе.

Атрибут *`fixed`* нежелательно использовать для добавления важной информации в атрибуты или элементы в XML-документе кроме случая, когда он используется для атрибута с указанием use=”*`required`*”.

5.3.2.3.3.2 Пояснение

Эти два атрибута позволяют понимающему схемы XML-процессору добавлять информацию в документ-экземпляр на основании информации из схемы. Во многих случаях это очень ценно. Например, это позволяет схеме вставить свой номер версии в документ-экземпляр, или указать способ кодировки текстового значения без нужды прописывать его в каждом документе-экземпляре.

Тем не менее, есть и недостатки:

- XML-процессор, не понимающий схемы, не сможет получить доступ к этой информации

- человек, читающий документ с данными, не догадывается, что нужно ещё посмотреть на схему, чтобы получить исчерпывающую информацию, и пр.

В общем случае преимущества вставки информации в документ с помощью определений в схеме может быть совмещено с опасностями.

5.3.2.3.4 Содержимое элементов

5.3.2.3.4.1 Рекомендация

Нежелательно допускать, чтобы необязательные элементы, которые спроектированы так, чтобы иметь содержимое, могли оказаться пустыми в XML-

документе. Желательно также, чтобы схема проверяла, что они либо отсутствуют, либо заполнены.

5.3.2.3.4.2 Пояснение

При наличии в схеме необязательных элементов отсутствие данных может быть выражено опусканием этого элемента из документа. Тем не менее, иногда оказывается проще указать элемент пустым, нежели не указывать его. Например, при сериализации информации из унаследованной системы или БД в XML, проще использовать один и тот же код для вывода заполненных и пустых XML-элементов. Тем не менее, это считается не столь важным, как обеспечение ясности и лаконичности на уровне интероперабельности (т.е., на уровне XML). Кроме того, пустые, но присутствующие элементы занимают излишние системные ресурсы.

Если элемент обязательно указывать, то, вероятно, бизнес-правила документа также требуют, чтобы он содержал данные. Если это так, то следует отразить этот факт в XML-схеме.

5.3.2.3.5 Локальные определения атрибутов против глобальных

5.3.2.3.5.1 Рекомендация

Как правило, желательно определять атрибуты локально в контексте элемента-владельца.

5.3.2.3.5.2 Пояснение

Это позволяет добиться простоты и читабельности схем. Если атрибут с похожим определением используется в разных местах, следует определить тип данных, или «группу атрибутов», чтобы обеспечить повторное использование. Если требуется более комплексное решение, то, вероятно, следует использовать элемент, а не атрибут для представления этих данных.

5.3.2.4Указания по RDF-совместимости XML-формата

С нарастанием интереса к Semantic Web и развитием его приложений, возрастает выгода от учёта соображений совместимости с RDF наравне с другими руководствами дизайна XML-документов и схем. Обеспечение «RDF-дружественности» данных является важным шагом для поддержки последних направлений развития Web-индустрии, которые могут найти отражение в будущих мировых стандартах и стандартах СЭВ и ИС ЭАР.

Гибкость RDF/XML-синтаксиса максимально упрощает приведение имеющихся XML-схем в соответствие со стилистикой RDF. Для обеспечения RDF-совместимости рекомендуется руководствоваться следующими указаниями при описании XML-схем.

5.3.2.4.1 Каждый элемент должен принадлежать к некоторому пространству имён

5.3.2.4.1.1 Рекомендация

Необходимо, чтобы любой используемый в XML-экземплярах элемент или атрибут принадлежали к некоторому пространству имён. Это означает, что все целевые схемы должны указывать targetNamespace, все локальные элементы и атрибуты должны быть квалифицируемы: `elementFormDefault` должен быть `qualified`. Использование атрибутов без префикса не рекомендуется, так как приводит к путанице с пространствами имён: `attributeFormDefault` желательно установить `qualified`.

5.3.2.4.1.2 Пояснение

RDF не допускает не-использования XML Namespaces. Это не столько ограничение, сколько разумное требование для обеспечения интероперабельности: XML без пространств имён является существенным недостатком при интеграции систем.

Это не значит, что все элементы должны иметь префикс пространства имён. Для удобства многие документы могут объявлять наиболее часто используемое пространство имён как «пространство имён по умолчанию» (default namespace), так что элементы из этого пространства имён указываются без префикса.

5.3.2.4.2 URI-идентификация и URI-ссылки

5.3.2.4.2.1 Рекомендация

Желательно идентифицировать все объекты данных, описываемые в XML, URI-идентификаторами (RFC 2396) и указывать ссылки на них опять же с помощью URI-идентификаторов.

Для указания URI используется атрибут `rdf:about`, для ссылки - URI указывается с помощью атрибута `rdf:resource` при пустом элементе.

5.3.2.4.2.2 Пояснение

URI-идентификация – крайне полезный механизм для интеграции распределённых данных. URI используется вместо primary key таблиц и прочих цифровых идентификаторов для исключения конфликта идентификаторов в контексте СЭВ и ИС ЭАР. URI-идентификатор может быть сформирован на основе такого primary key или иного ключа, но обязательно с префиксом, указывающим «пространство имён» данного ключа, например указывающим организацию-поставщика данных.

5.3.2.4.2.3 Примеры

```
<Organization rdf:about="http://economy.gov.ru/">
```

```
  <dc:title> Минэкономразвития РФ </dc:title>
  <acronym> МЭРиТ </acronym>
  <foundationDate>1967-08-13</foundationDate>
  <dateClosedOut>1967-08-13</dateClosedOut>
</Organization>

<Department rdf:about="http://sample.ru/namespaces/departments/54321">
  <dc:title> Подразделение X </dc:title>
  <foundationDate>1967-08-13</foundationDate>
  <dateClosedOut>1967-08-13</dateClosedOut>
  <organization rdf:resource="http://economy.gov.ru/"/>
</Department>

<Department rdf:about="http://sample.ru/namespaces/departments/54322">
  <dc:title> Подразделение Y подчинённое подразделению X </dc:title>
  <organization rdf:resource="http://economy.gov.ru/"/>
  <!-- ссылка на подразделение X: -->
  <superDepartment rdf:resource="http://sample.ru/namespaces/departments/54321"/>
</Department>

<Person rdf:about="http://sample.ru/namespaces/persons/12345">
  <vCard:N>
    <vCard:Given> Иван </vCard:Given>
    <vCard:Other> Иванович </vCard:Other>
    <vCard:Family> Иванов </vCard:Family>
  </vCard:N>
  <job>
    <!-- должность - начальник по ОК 016-94 -->
    <jobTitle rdf:resource="http://gov.ru/namespaces/classifiers/OK-016-94/21447"/>
    <!-- ссылка на подразделение X: -->
    <worksFor rdf:resource=" http://sample.ru/namespaces/departments/54321"/>
  </job>
</Person>
```

5.3.2.4.3 Совместимость XML-структуры с вариантами синтаксиса RDF/XML

Желательно, чтобы XML-формат, задаваемый XML-схемой, следовал RDF/XML-правилам представления концептуальной модели данных в XML-виде.

Ориентированный на данные XML-документ имеет строгую структуру, не допускающую перемежения текста с элементами (mixed content), на самом деле отражающую концептуальную модель предметной области приложения. Одни XML-

элементы описывают объекты (например PurchaseOrder в примере из XML Schema Primer), их атрибуты и вложенные элементы – описывают свойства этих объектов и их отношения с другими объектами (например, shipTo, billTo, orderDate, item). Свойства могут содержать примитивные значения, такие как строки, даты или числа, либо структурированные значения – то есть, тоже объекты. Соответственно, внутри такого тега-свойства указывается описание свойств этих вложенных объектов, и т.д. Могут быть также группирующие элементы.

RDF/XML имеет гибкий синтаксис, допускающий различные варианты записи RDF-данных в XML. Эти варианты позволяют **почти любой XML** проинтерпретировать как RDF-данные. Тем не менее, следует пройти эти варианты, чтобы убедится в совместимости XML-формата с синтаксисом RDF/XML:

- каждый элемент первого уровня описывает объект (ClassElt), и его имя тега считается названием класса описываемого объекта (типом объекта)

```
<purchaseOrder ...>
...
</purchaseOrder>
```

- каждый элемент, описывающий объект, не должен содержать внутри себя текст (xsd:complexType, mixed=”false”)
- он может содержать атрибуты или вложенные элементы, описывающие свойства этого объекта (PropertyElt)

```
<purchaseOrder orderDate="1999-10-20">
    <shipTo>
      ...
    </shipTo>
    <billTo>
      ...
    </billTo>
    <comment>Hurry, my lawn is going wild!</comment>
    ...
</purchaseOrder>
```

- если свойство примитивного типа, то его значение указывается как текст внутри элемента-свойства

```
<comment>Hurry, my lawn is going wild!</comment>
```

- язык текста может быть указан с помощью xml:lang

```
<comment xml:lang=”en”>Hurry, my lawn is going wild!</comment>
```

- если свойство объектного типа, то

- либо объект указывается ссылкой – по уникальному в пределах документа ID с помощью *rdf:nodeID*, или по URI с помощью *rdf:resource*. Тогда элемент должен быть пустым.

```
<job>
  <jobTitle> директор </jobTitle>
  <worksFor rdf:resource="http://economy.gov.ru/samples/232323"/>
</job>
```

- либо объект описывается внутри тега-свойства.
- либо внутри тега-свойства вкладывается тег-объект (ClassElt), а внутри тега-объекта описываются его свойства

```
      <worksFor>
        <Organization>
          <dc:title> Минэкономразвития РФ </dc:title>
        </Organization>
      </worksFor>
```

- либо тег-объект не указывается и свойства объекта перечисляются непосредственно внутри тега-свойства.
  Замечание: при этом нельзя указать атрибут rdf:about.

```
<shipTo country="US">
    <name>Alice Smith</name>
    <street>123 Maple Street</street>
    <city>Mill Valley</city>
    <state>CA</state>
    <zip>90952</zip>
</shipTo>
```

- если для одного свойства указано несколько значений, то элемент-свойство повторяется, и каждое значение указывается в отдельном теге.

```
    <ex:Task>

      ...
      <ex:performer>
     <Department rdf:about="http://sample.ru/dep1">
            <dc:title> Отдел X Минэкономразвития РФ </dc:title>
     </Department>
      </ex:performer>
      <ex:performer>
     <Person rdf:about="http://sample.ru/per1">
    <vCard:N>
```

```
      <vCard:Given> Иван </vCard:Given>
      <vCard:Other> Иванович </vCard:Other>
      <vCard:Family> Иванов </vCard:Family>
    </vCard:N>
     </Person>
      </ex:performer>
    </ex:Task>
```

- Следует быть крайне внимательным при использовании элементов-контейнеров (группирующих несколько значений). Применение таких элементов не рекомендуется. Тем не менее, допускается применение элементов-контейнеров с семантикой «коллекции значений». Это означает, что тег-контейнер считается тегом-свойством (PropertyElt), а вложенные в него теги – тегами-объектами (ClassElt). Соответственно, имя тега-контейнера должно быть именем свойства, а имя тега-объекта – именем класса:

```
    <ex:Task>

      ...
      <ex:performers>
      <Department>
             <dc:title> Отдел X Минэкономразвития РФ </dc:title>
      </Department>
      <Person>
    <vCard:N>
      <vCard:Given> Иван </vCard:Given>
      <vCard:Other> Иванович </vCard:Other>
      <vCard:Family> Иванов </vCard:Family>
    </vCard:N>
      </Person>
      </ex:performers>
    </
 ex:Task>
```

Пример недопустимого использования: attachment не удовлетворяет определению ClassElt, так как имеет «простой» тип - xsd:simpleType.

```
<email>
  <from>some@mail.ru</from>
  <to>other@mail.ru</to>
```

```
  <msgSubject>Dinner tonight</msgSubject>
  <attachments>
    <attachment>data\sample1.txt</attachment>
    <attachment>data\sample2.txt</attachment>
  </attachments>
  <cc>frank@snee.com</cc>
</email>
```

Другой пример некорректного использования: элемент contact-info используется для тематической группировки свойств объекта, в RDF это как правило соответствует понятию суперсвойства. Следует опускать такие группирующие элементы:

```
<Organization>
  ...
  <contact-info>
    <email>some@mail.ru</email>
    <phone>(095)1234567</phone>
    <address>Москва, Кремль</address>
  </contact-info>
</Organization>
```

5.3.2.4.4 Применение элементов стандартизованных схем

5.3.2.4.4.1 Рекомендация

Желательно использовать термины (классы и свойства), определённые в стандартизованных в Semantic Web обменных схемах. В контексте XML-схем это соответствует использованию этих терминов в качестве названий (QName) элементов, атрибутов, типов в схеме.

5.3.2.4.4.2 Пояснение

В настоящее время заметна широкая тенденция по стандартизации RDF-словарей свойств метаданных для конкретных предметных областей – так называемых «обменных схем», или «профилей метаданных». Использование терминов (свойств, словарей значений и пр.), зафиксированных в стандартах, позволяет приложениям легко интегрироваться между собой, обмениваться информацией, понятной им всем. Например, при получении данных из сторонней системы, приложение может найти среди неизвестных ему свойств некоторые свойства, регламентированные стандартом, и соответственно будет уверено в их смысле, семантике, сможет правильно их проинтерпретировать. Это называется «семантической интероперабельностью», и считается одним из основных преимуществ Semantic Web.

Dublin Core Metadata Initiative (DCMI) определил минимальный набор свойств для описания цифровых ресурсов Web (DCES: ISO 15836), а также их детализацию в рамках «общего профиля» [12]. Отдельные рабочие группы DCMI занимаются стандартизацией более специализированных профилей метаданных таких предметных областей, как библиотечная информация, образование, правительственная сфера, информация о людях и пр.

Dublin Core стал базисом для других «стандартов обмена». В первую очередь, следует упомянуть стандарт Publishing Requirements for Industry Standard Metadata (PRISM) [45], разработанный издательскими организациями для обмена метаданными о публикациях (документах, журналах, книгах и пр.). Государственный архив Австралии выдвинул и стандартизовал основанный на Dublin Core набор профилей метаданных для описания государственной информации – AGLS Metadata Standard [23]. Широкое применение нашли предложения по представлению информации стандарта VCard («визитная карточка») в RDF [14]. VCard определяет свойства для описания информации о людях, их контактной информации и пр.

5.3.2.4.5 Словари с URI-идентификацией вместо enumerated-типов

5.3.2.4.5.1 Рекомендация

Желательно указывать элементы контролируемого словаря с помощью URI через rdf:resource, а не использовать enumerated-типы XML-схемы.

5.3.2.4.5.2 Пояснение

В XML Schema вошло в традицию (отчасти, ввиду отсутствия альтернативы) использование перечислимых типов данных для определения контролируемых словарей, таких как единицы измерения, типы валют, названия/коды стран и языков и пр. Например:

```
<country>RU</country>
```

Этот подход имеет очевидные недостатки, такие, как огромные размеры схем для больших словарей, невозможность динамического расширения словарей, сложность интеграции с другими системами, которые выбрали для тех же целей другой набор значений перечислимого типа, и пр. Напротив, в RDF такие словари описывают стандартными средствами, как RDF-ресурсы, идентифицируемые URI, обладающие собственными свойствами (например, с каждым языком может быть связана информация о его названии и переводах названия, 2-буквенных и 3-буквенных ISO-кодах и пр.). Для указания элемента RDF-словаря как значения некоторого свойства используется rdf:resource-ссылка по URI. Например, следующим образом указывается страна:

```
<country rdf:resource="http://prismstandard.org/vocabs/ISO-3166/RU"/>
```

5.4 Регистрация сервисов и метаинформации

5.4.1 Реестр сервисов и метаинформации

Если предполагается сообщение информации о Web-сервисах («публикация») для последующего их поиска, BasicProfile-1.0 от WS-I рекомендует использовать UDDI для описания разработчиков Web-сервиса и предоставляемых им услуг. Описание области применения, предполагаемого применения и типа Web-сервиса выполняется в терминах UDDI, а подробное техническое описание выполняется в терминах WSDL. Если оба этих описания используются и предоставляют перекрывающиеся сведения, BasicProfile-1.0 специфицирует, что эти описания не должны противоречить друг другу.

Регистрация всех аспектов Web-сервиса в регистрах UDDI не является обязательной. Не для всех сценариев использования требуются указания метаданных и сведений для поиска, обеспечиваемые UDDI, но если такие указания все же необходимы, то их следует предоставлять именно через UDDI.

Заметим, что Web-сервисы, входящие в UDDI V2, не вполне согласуются с BasicProfile-1.0, т.к. они не могут принимать сообщения в кодировках UTF-8 и UTF-16, как это требуется в данном Профиле (в них допускается лишь UTF-8). Это разногласие довольно естественно, учитывая тот факт, что UDDI V2 был разработан и достаточно широко использовался до представления BasicProfile-1.0. Разработчикам UDDI известно об этом недостатке UDDI V2 и они учтут это в своей последующей работе.

Соответствующий раздел BasicProfile-1.0 включил в себя с некоторыми расширениями следующие спецификации:

- UDDI Version 2.04 API Specification, Dated 19 July 2002
- UDDI Version 2.03 Data Structure Reference, Dated 19 July 2002
- UDDI Version 2 XML Schema

5.4.2 bindingTemplates

В соответствующем разделе BasicProfile-1.0 используются следующие спецификации (некоторые их разделы):

- UDDI Version 2.03 Data Structure Reference, Section 7

UDDI представляет экземпляры Web-сервисов как элементы uddi:bindingTemplate , при этом uddi:bindingTemplate играет роль, аналогичную wsdl:port, но предоставляет некоторые возможности, невыразимые в WSDL. Для поддержания совместимости WSDL-

и UDDI-описаний, BasicProfile-1.0 формулирует следующие ограничения на конструкцию элементов uddi:bindingTemplate.

WSDL-элемент soapbind:address требует непосредственного указания сетевого адреса экземпляра сервиса. С другой стороны, UDDI V2 предлагает две возможности указания сетевого адреса представляемого в нем сервиса. Первая - uddi:accessPoint – соответствует методу WSDL прямого указания адреса. Вторая - uddi:hostingRedirector - предоставляет ориентированный на Web-сервис не непосредственный механизм получения адреса, не согласующийся с подходом WSDL.

R3100 REGDATA (регистровые данные) типа *uddi:bindingTemplate*, представляющие соответствующий стандарту экземпляр сервиса, ДОЛЖНЫ содержать элемент *uddi:accessPoint*.

Например,

НЕПРАВИЛЬНО:

```
<bindingTemplate bindingKey="...">
   <description xml:lang="EN">BarSOAPPort</description>
   <hostingRedirector bindingKey="..."/>
   <tModelInstanceDetails>
      ...
   </tModelInstanceDetails>
</bindingTemplate>
```

ПРАВИЛЬНО:

```
<bindingTemplate bindingKey="...">
   <description xml:lang="EN">BarSOAPPort</description>
   <accessPoint>http://example.org/myBarSOAPPort</accessPoint>
   <tModelInstanceDetails>
      ...
   </tModelInstanceDetails>
</bindingTemplate>
```

5.4.3 Элементы tModel

В соответствующем разделе BasicProfile-1.0 используются следующие спецификации (некоторые их разделы):

- UDDI Version 2.03 Data Structure Reference, Section 8

UDDI представляет типы Web-сервисов как элементы uddi:tModel (см. UDDI Data Structures section 8.1.1.). Эти элементы могут, хотя и не обязаны, указывать (используя URI) на документ, содержащий сами спецификации. В общем случае UDDI игнорирует механизм описания типов Web-сервисов. BasicProfile-1.0 не может так же игнорировать этот механизм, т.к. интероперабельность будет чрезвычайно затруднена, если отсутствует описание типов Web-сервисов или оно приводится в произвольной форме.

UDDI API Specification, appendix I.1.2.1.1 позволяет (но не требует), чтобы элементы uddi:tModel , использующие WSDL для описания типов Web-сервисов, указывали, что они используют WSDL в качестве языка представления этих описаний. Но если опустить это указание, то возникают проблемы интероперабельности, ввиду неясности, какой именно язык использован для описаний.

В связи с этим BasicProfile-1.0 формулирует следующие ограничения на конструкцию элементов uddi:tModel, описывающих типы Web-сервисов:

BasicProfile-1.0 выбирает в качестве языка описаний WSDL, т.к. это наиболее широко используемый язык в данной области.

R3002 REGDATA (регистровые данные) типа uddi:tModel, представляющие соответствующий стандарту тип Web-сервиса, ДОЛЖНЫ содержать использовать в качестве языка описаний WSDL.

Для указания того, что соответствующий стандарту тип Web-сервиса использует WSDL, BasicProfile-1.0 принимает формулировку UDDI.

R3003 REGDATA (регистровые данные) типа uddi:tModel, представляющие соответствующий стандарту тип Web-сервиса, ДОЛЖНЫ быть категоризированы с использованием в конструкции uddi:types с указанием категории "wsdlSpec".

Для того чтобы конструкция uddi:overviewURL в uddi:tModel была разрешима в wsdl:binding, BasicProfile-1.0 должен предложить некоторое соглашение, обеспечивающее распознавание различных wsdl:binding в WSDL-документе. Таковым наиболее широко признанным соглашением является «The UDDI Best Practice for Using WSDL in a UDDI Registry» (Рекомендации UDDI по Использованию WSDL в Регистре UDDI).

R3010 REGDATA (регистровые данные) типа uddi:tModel, представляющие соответствующий стандарту тип Web-сервиса, ДОЛЖНЫ соответствовать V1.08 of the UDDI Best Practice for Using WSDL in a UDDI Registry.

Для предотвращения рассогласований, wsdl:binding, указываемый в uddi:tModel, должен соответствовать требованиям BasicProfile-1.0.

R3011 wsdl:binding, указываемый REGDATA типа uddi:tModel, ДОЛЖЕН соответствовать BasicProfile-1.0.

5.5 Безопасность

В данном разделе в дополнение к общим вопросам поддержки безопасного общения ИС посредством технологии Web-сервисов даны дополнительные рекомендации по выбору безопасного протокола передачи SOAP-сообщений в сети Интернет.

Как и для всех сетевых информационных технологий, обеспечение безопасности критично для Web-сервисов. В Web-сервисах, как и в других информационных технологиях, обеспечение безопасности состоит в понимании потенциальных опасностей, которые представляет атакующий субъект, и принятии операционных, физических и технологических контрмер для снижения риска успешной атаки до допустимого уровня. Но поскольку представление о «допустимом уровне риска» значительно различается в разных приложениях, а также весьма различна, стоимость контрмер, невозможно предложить универсальное «правильное решение» по обеспечению безопасности Web-сервисов. В каждом конкретном случае следует принимать решение о корректной сбалансированности между степенью контрмер и допустимым риском.

Учитывая вышесказанное, можно, тем не менее, указать некоторый общий подход к выбору контрмер, который, как показывает опыт, обеспечивает приемлемый уровень риска для большинства Web-сервисов. BasicProfile-1.0 рекомендует, но не выдвигает как обязательные, следующие широко распространенные средства:

- RFC2818: HTTP Over TLS
- RFC2246: The TLS Protocol Version 1.0

  Расширения:
  - TLS Cyphersuite – TLS допускает использование любых алгоритмов шифровки.
  - TLS Extensions – TLS допускает дополнительные проверки в фазе контакта.

- The SSL Protocol Version 3.0

  Расширения:
  - SSL Cyphersuite – TLS допускает использование любых алгоритмов шифровки.

- RFC2459: Internet X.509 Public Key Infrastructure Certificate and CRL Profile

  Расширения:
  - Certificate Authority – выбор Certificate Authority оставляется на усмотрение участвующих сторон (как частное соглашение).
  - Certificate Extensions – X509 допускает произвольные расширения в представлении сертификата.

Иначе говоря, соответствующие стандарту Web-сервисы могут использовать HTTPS, могут применять другие технологии безопасности или вообще ничего не применять.

HTTPS широко признан как наиболее зрелый стандарт обеспечения базового уровня конфиденциальности для шифруемых транспортных соединений. Таким образом, HTTPS представляется первоочередным и наиболее простым средством обеспечения основного уровня безопасности для большинства реальных приложений Web-сервисов. HTTPS может также использоваться для обеспечение аутентификации клиента по предоставляемым им сертификатам.

5.5.1 Использование HTTPS

HTTPS является настолько полезным и широко признанным механизмом обеспечения базовой безопасности, что это приводит BasicProfile-1.0 к разрешению его использования:

R5000 Экземпляр Web-сервиса МОЖЕТ требовать использования HTTPS.

R5001 Если некоторый экземпляр Web-сервиса требует использования HTTPS, атрибут расположения элемента soapbind:address в описании wsdl:port ДОЛЖЕН представлять собой URI со схемой "https" или со схемой"http".

Стандартное использование HTTPS обеспечивает аутентификацию экземпляра Web-сервиса пользователем, но не аутентификацию пользователя этим экземпляром. Во многих случаях такое ограничение представляется слишком рискованным для

проведения операции. Включение в BasicProfile-1.0 средств взаимной аутентификации HTTPS позволяет экземплярам Web-сервиса выполнять дополнительную контрмеру аутентификации пользователя. В случаях, когда аутентификация экземпляра пользователем считается недостаточной, эта дополнительная возможность снижает риск в степени, достаточной для проведения операции:

R5010 Экземпляр Web-сервиса МОЖЕТ требовать использования HTTPS со взаимной аутентификацией.

5.6 Транзакции

5.6.1 Описание расширения структуры стандартных SOAP-сообщений

5.6.1.1 Понятие контекста транзакции с участием WEB-сервисов

Web-сервисы все более широко используются как интегрирующая платформа для распределенных enterprise-приложений. Результирующие приложения могут быть очень сложны по структуре, со сложными связями между составляющими их приложениями, обусловленными необходимостью распространять общие данные или контекст между взаимодействиями. Более того, исполнение такого приложения может занимать долгое время, и может включать продолжительные периоды неактивности, часто обусловленные тем, что составляющее приложения требуют взаимодействия с пользователем.

Возможность объединять произвольные единицы распределенной работы – требование по многим аспектам интеграции распределенных приложений, таких как быстрые обновления информации, потоки работ, b2b взаимодействия, автоматизированные бизнес-процессы и другие. Объединяя работу в рамках общего контекста, позволяет участникам процесса узнавать, находятся ли они в пределах одной и той же бизнес-задачи. Для того, чтобы коррелировать работу распределенных участников в пределах одной бизнес-задачи, необходимо распространять дополнительную информацию (*контекст*) между участниками. Контекст содержит информацию, такую как уникальный идентификатор, который позволяет серии операций использовать общий выход. Например, заголовок SOAP-сообщений может содержать контекстную информацию, которая распространяется при взаимодействии с сервисом, или при обмене SOAP-сообщениями большого числа участников для того, чтобы создать поток работ или другую композицию Web-сервисов. Информация бизнес контекста может

представляться как Web-ресурс, доступный посредством URI. Контекстные операции могут прозрачно для приложения распространяться с нормальными сообщениями, рассылаемыми участникам во время выполнения Web-сервиса, или это может быть осознанное действие, инициируемое участником бизнес-задачи. Один и тот же тип бизнес контекста не всегда применим во всех случаях. Таким образом, он должен быть расширяем характерным для приложения способом: сервисы должны иметь возможность наращивать контекст так, как им требуется. Более того, может понадобиться, чтобы к этим участникам или сервисам поступал дополнительный характерный для приложения контекст. Таким образом, распространение бизнес-контекста является фундаментальным требованием для многих распределенных систем, включая Web-сервисы.

5.6.1.2Описание общей структуры контекста бизнес-транзакции, передаваемого в заголовках SOAP-сообщений

Действие (Activity) представляет собой единицу распределенной работы, включающей одного или более участников (сервисы, компоненты, объекты). Действие создается, выполняется и затем завершается. Выход действия – это результат его завершения, который может использоваться для определения последующего управляющего потока, и может привести к генерации сообщения об ошибке. Действия могут выполняться длительные промежутки времени (минуты, часы, дни, …). Во время жизни действию могут требоваться периоды координации между его участниками или может потребоваться взаимодействие с другими сервисами, такими как управление транзакциями и безопасность.

Для того, чтобы действие могло охватывать набор Web-сервисов, необходимо распространять между участниками дополнительную информацию – контекст. Сервис распространения контекста должен определять структуру бизнес-контекста, которая позволила бы действию быть уникально идентифицированным таким образом, чтобы работа могла быть коррелированна. Более того, сервис контекста должен давать возможность приложениям и сервисам расширять структуру контекста характерным для приложения образом (для координации, транзакций, безопасности или репликации). Стандартная структура бизнес-контекста представлена в следующем примере:

```
<xs:complexType name="ContextType">
  <xs:sequence>
    <xs:element name="context-identifier" type="xs:anyURI"/>
    <xs:element name="activity-service" type="xs:anyURI" minOccurs="0"/>
```

```
      <xs:element name="type" type="xs:anyURI" minOccurs="0"/>
      <xs:element name="activity-list" minOccurs="0">
        <xs:complexType>
          <xs:sequence>
            <xs:element name="service" type="xs:anyURI" minOccurs="0"
            maxOccurs="unbounded"/>
          </xs:sequence>
          <xs:attribute name="mustUnderstand" type="xs:boolean" use="optional"
          default="false"/>
          <xs:attribute name="mustPropagate" type="xs:boolean" use="optional"
          default="false"/>
        </xs:complexType>
      </xs:element>
      <xs:element name="child-contexts" minOccurs="0">
        <xs:complexType>
          <xs:sequence>
            <xs:element name="child-context" type="tns:ContextType"
            maxOccurs="unbounded"/>
          </xs:sequence>
        </xs:complexType>
      </xs:element>
      <xs:any namespace="##other" processContents="lax" minOccurs="0"
      maxOccurs="unbounded"/>
    </xs:sequence>
    <xs:attribute name="timeout" type="xs:int" use="optional"/>
</xs:complexType>
```

Контекст состоит из следующих элементов:

- Необходимый URI-идентификатор называемый идентификатор контекста. Он гарантирует глобальную уникальность отдельному действию (такой идентификатор можно рассматривать так же и как «корреляционный» идентификатор или значение, используемое для того, чтобы выявить принадлежность задания одному бизнес-действию);
- Опциональный элемент activity-service, который идентифицирует сервис контекста, отвечающий за генерацию контекста;
- Опциональный идентификатор, указывающий тип действия;
- Опциональный список сервисов, участвующих в бизнес-действии к настоящему моменту, называемый participating-services;

- Опциональный список дочерних действий в элементах child-contexts;
- Значение таймаута, которое время жизни контекстной информации. По окончании этого периода, структура действия, на которое она ссылается, перестает существовать (т.е. действие могло завершиться или произошла очистка занятых ресурсов).

Распространение контекста так же может происходить посредством протоколов, отличных от тех, которые использует приложение. Сервис контекста не определяет стандартных средств, с помощью которых происходит распространение бизнес-контекста, вынося этот вопрос на уровень реализации.

5.6.2 Рекомендации по реализации координирующих протоколов бизнес-транзакций на базе средств обработки исключительных ситуаций и компенсации языка описания АРП BPEL4WS

Бизнес процесс определяет потенциальный порядок исполнения операций набора Web-сервисов, данные, которыми они обмениваются, набор вовлеченных партнеров взаимодействия и регламент их взаимодействия, обработку исключительных ситуаций и другие аспекты взаимодействия Web-сервисов. Данные механизмы позволяют определять долгоживущие транзакции на маршруте процесса, увеличиваю надежность и целостность данных в приложениях, использующих технологию Web-сервисов.

Язык описания автоматизированных бизнес-процессов Business Process Execution Language For Web Services (BPEL4WS или кратко BPEL) позволяет определять бизнес-процессы и их связь с внешними Web-сервисами. Это включает определение того, как бизнес процесс использует Web-сервисы для достижения своей цели, равно как и определение Web-сервисов, предоставляемых бизнес-процессом. Бизнес процесс, определяемый BPEL4WS, является полностью исполнимым и переносимым между различными средами исполнения бизнес-процессов. Бизнес-процесс взаимодействует с Web-сервисами партнеров, в независимости от того, реализуются ли эти сервисы посредством BPEL4WS или нет. Наконец, BPEL4WS поддерживает определение бизнес протоколов между партнерами представления сложных внутренних бизнес-процессов.

BPEL4WS-процесс взаимодействует с WSDL-портами, которые могут вернуть сообщение об ошибке процессу. Более того, процесс сам может выявить ошибочную ситуацию, которая ведет к внутренним ошибкам. BPEL4WS предоставляет механизм, который позволяет обрабатывать подобные исключительные ситуации.

Центральной идеей данных механизмов являются так называемые *обработчики исключений,* которые могут перехватывать и обрабатывать ошибки. В фрагменте кода ниже приведен пример описания обработчика исключений, который определяет набор исключений, которые он собирается обрабатывать посредством соответствующего набора элементов *<catch>* (строка 36). Внутри таких элементов может присутствовать действие любого типа. Это действие будет выполнено, когда возникнет соответствующее исключение. Обработчик исключений перехватывает ошибку *noSeatsAvailable*, возвращаемую участником процесса. Когда происходит исключение, соответствующее сообщение об отмене операции отправляется участнику посредством вложенного действия *<invoke>* (строки с 37 по 40).

```
35 <faultHandlers>
36    <catch  faultName="noSeatsAvailable">
37       <invoke  partner="customer"
38                portType="travelPT"
39                operation="sendRejection"
40                inputContainer="rejection"/>
41    </catch>
42 </faultHandlers>
```

Как правило, реакции на исключительные ситуации, такие как ошибки, зависят от того, на какой стадии исполнения уже находится процесс. В BPEL4WS-процессе это может определяться посредством так называемых *контекстов*. Контекст (scope) представляет собой структурированное действие, которое позволяет группировать другие действия. В дополнение к этому, оно позволяет определять общий контекст исполнения соответствующего набора вложенных в него действий (обработчики ошибок, которые перехватывают исключения, возникающие при выполнении одного из действий, входящих в контекст).

Когда исключение происходит в контексте, обычное выполнение в нем прерывается, и исключение передается обработчикам исключений. Действие, заключенное в соответствующем обработчике ошибок, пытается исправить ситуацию таким образом, чтобы могло быть продолжено нормальное выполнение процесса вне контекста или была выбрана альтернативная ветви маршрута процесса.

Все это может потребовать отмены действий, которые уже были завершены в пределах контекста. Например, если билет, необходимый для поездки, не имеется в наличии, резервирование места в отеле и машины, которое было сделано ранее, должно

быть отменено. Действия, требуемые для отмены уже завершенных действий, называются компенсаторами. Обработчик ошибок контекста может использовать компенсаторы для того, чтобы отменить действия, произведенные внутри контекста. Если исключительная ситуация не может быть исправлена, обработчик ошибок пропустит исключение дальше или сгенерирует новое, которое в итоге будет перехвачено обработчиком исключений охватывающего контекста.

Таким образом, BPEL4WS позволяет определять наборы действий, которые могут быть коллективно отменены при возникновении исключений посредством механизма контекстов. Такой набор действий представляет собой некоторую единицу работы, некоторую транзакцию. Действия внутри контекста либо выполняются все, либо все компенсируются. В противоречие этому хорошо известные традиционные транзакции (например, транзакции в базах данных) реализуются посредством блокировок, предоставляющих ресурсы в монопольный доступ транзакции на период ее выполнения. Это предполагает, что транзакции определяют короткоживущие единицы работы, и что блокировки могут быть быстро сняты. Поскольку контексты BPEL4WS как правило имеют долгий период жизни, на практике не используется механизм блокировки ресурсов; вместо него используются компенсаторы. Это позволяет снимать блокировки сразу после выполнения действия, но приходится определять логику компенсации для отмены уже выполненных действий. Результирующие единицы работы или транзакции называются *долгоживущими транзакциями* (*long running transactions).*

Долгоживущие транзакции в BPEL4WS определяются контекстами, и контексты поддерживают вложение. Существует протокол согласования между контекстом и его родительским контекстом, позволяющий определить выход долгоживущей транзакции, представленной контекстом. Соответствующий протокол описан в спецификации WS-Transaction. В то время как долгоживущие транзакции в BPEL4WS предполагают, что контекст и все его вложенные контексты содержатся внутри единого процесса и обрабатываются одним BPEL4WS-процессором, протокол согласования WS-Transaction этого не предполагает. Таким образом, дальнейшее расширение BPEL4WS может поддержать долгоживущие транзакции, которые являются распределенными не только на уровне процессов, но даже и на уровне BPEL4WS-процессоров.

WS-Transaction так же определяет протоколы для координации распределенных атомарных транзакций. Дальнейшее расширение BPEL4WS может поддержать распределенные атомарные транзакции, состоящие из действий одного процесса или даже разных процессов.

BPEL4WS представляет собой пласт над WSDL – это означает, что BPEL4WS использует WSDL для того, чтобы описать действия, которые присутствуют в бизнес-процессе, и для того, чтобы описать Web-сервисы, предоставляемые бизнес-процессом. WS-Transaction описывает как протокол для модели долгоживущих транзакций, определенной в BPEL4WS, так и атомарные транзакции между обычными Web-сервисами.

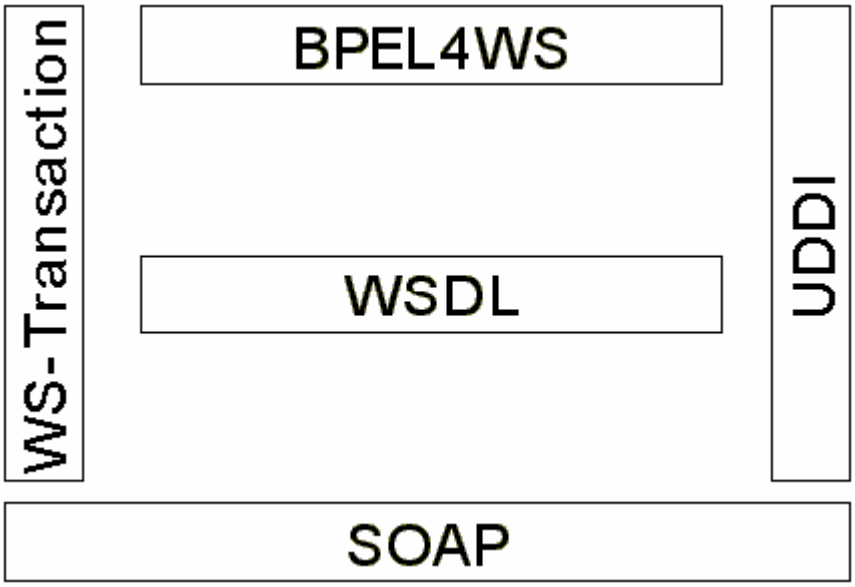


Рисунок 20. Взаимосвязь WEB-стандартов при описании рабочих процессов

5.7 Рабочие процессы

Дополнительных общих рекомендаций по оформлению BPEL4WS процессов в рамках конкретных СУВ помимо тех, которые описаны в разделе «4.7 Рабочие процессы» и приведены в спецификации BPEL4WS, не предъявляется.

Рекомендации по использованию конструкция BPEL4WS для обработки исключительных ситуаций на маршруте процесса приведены в разделе «5.6.2 Рекомендации по реализации координирующих протоколов бизнес-транзакций на базе средств обработки исключительных ситуаций и компенсации языка описания АРП BPEL4WS».

5.8 Глобальная идентификация

Поскольку Handle System не предоставляет возможности поиска по метаданным идентификатора, для взаимного отображения локальных идентификаторов в глобальные потребуется сделать следующее:

1. завести собственное пространство имён для каждой информационной системы, которая содержит информацию о ресурсе вместе с локальными для этой системы идентификаторами;
2. завести пространство имён для глобальных идентификаторов;
3. для каждого ресурса завести Handle в глобальном пространстве имён;
4. для каждого локального идентификатора завести Handle в соответствующем локальном пространстве имён. Среди полей этого Handle добавить поле с типом “URN”, значением которого будет является глобальный идентификатор. В Handle глобального идентификатора добавить поле типа “URN”, значением которого будет являться локальный идентификатор.

После этого для отображения глобального идентификатора в локальные потребуется разрешить соответствующий глобальный Handle, и получить значения его полей с типом “URN”. Наоборот, для преобразования локального идентификатора в глобальный, необходимо разрешить соответствующий Handle в локальном пространстве имён, и прочитать поле с типом “URN”.

6 Рекомендации по использованию стандартов в рамках СЭВ

Основополагающими WEB-стандартами, поддерживаемыми программными средствами СЭВ, являются:

- ***W3C Web Services Description Language (WSDL) 1.1***
  Описание интерфейсов WEB-сервисов, набора их операций с описанием входных и выходных параметров-сообщений и их типов, отображения сообщений в транспортный протокол
- ***W3C Simple Object Access Protocol (SOAP) 1.1***
  Используется для удаленного вызова операций WEB-сервисов.
- ***W3C XML Schema***
  Применяется как основное средство для описания формата передаваемых по протоколу SOAP параметров операций WEB-сервисов. Так же служит для описания ключевых данных СЭВ для возможности интеграции с внешними системами в рамках канонической модели данных.
- ***Business Process Execution Language for Web Services Version 1.1***
  Используется для декларации описания бизнес-процессов, реализующих процедуры электронных взаимодействий, а так же как средство описания бизнес-транзакций на маршруте ПЭВ.
- ***OASIS UDDI Version 2.04***
  Используется как модель описания каталога организаций и сервисов, а так же для реализации интерфейсов для доступа к ним.

В данном разделе освещается специфика использования данных стандартов в рамках СЭВ. Прежде всего, указывается поддерживается ли и в каком объеме та или иная спецификация, указываются характерные особенности и ограничения по ее применению. При этом данные ограничения могут быть обусловлены как архитектурой СЭВ, так и текущей реализацией интеграционных механизмов.

6.1 ИС и сервисы

Среда Электронного Взаимодействия не накладывает дополнительных требований на использование стандарта W3C Web Services Description Language для описания

интерфейсов Web-сервисов Среды помимо тех рекомендаций, которые описаны в пункте «5.1 ИС и сервисы». При этом не вносятся дополнительных расширений в спецификацию.

6.2 Транспорт и сообщения

6.2.1 Рекомендации по способам инкапсуляции SOAP-сообщений в протокол HTTP в рамках СЭВ

6.2.1.1 Обоснование выбора протокола HTTP для передачи SOAP сообщений в рамках СЭВ

В работах 2004-ого года предполагается использовать только протоколы HTTP/HTTPS для транспорта сообщений SOAP. Такое решение было принято вследствие того, что данные протоколы позволяют решить все необходимые задачи и при этом достаточно просты, и широко используемы. Поддержка других протоколов приведет к необоснованному росту расходов на разработку. Также WS-I Basic Profile 1.0 рекомендует использование именно HTTP/HTTPS для транспорта.

6.2.1.2 Используемые способы HTTP-привязки SOAP сообщений в рамках СЭВ

Для привязки SOAP-сообщений к протоколам HTTP/HTTPS в рамках СЭВ будет использоваться стандартная привязка посредством HTTP-метода POST для передачи запроса и ответа сервиса в теле POST-запроса (см. рекомендации по POST-привязке SOAP-сообщений в разделе «5.2.1.1 Инкапсуляция в протокол HTTP»).

6.2.2 Рекомендации по способам инкапсуляции SOAP-сообщений в протокол SMTP в рамках СЭВ

6.2.2.1 Инкапсуляция SOAP-сообщений в протокол SMTP (X.400)

Для привязки SOAP-сообщений к протоколам SMTP (X.400) в рамках СЭВ будет использоваться привязка посредством вставки SOAP-сообщения в тело SMTP-сообщения. Запрос идентифицируется строкой в поле Subject, то есть инициатор запроса, отправляя сообщение, устанавливает идентификатор в поле Subject, в ответ сервис отправляет сообщение с таким же идентификатором в поле Subject.

6.2.2.2 Обоснование отклонения протокола SMTP для передачи SOAP сообщений в рамках СЭВ

Ввиду сложности реализации и возникновения необоснованных расходов протокол SMTP для передачи SOAP-сообщений в рамках работ 2004 года использоваться не будет. Основными недостатками данного протокола по сравнения с использованием протокола HTTP являются:

- трудность организации гарантированной доставки сообщений (отсылки и получения уведомлений о доставке);

- неинтегрирован со стандартными механизмами безопасности (по сравнению с протоколом HTTPS);
- менее широко используется для обмена SOAP-сообщениями между информационными системами из-за сложности интеграции.

6.3 Каноническая модель данных и язык спецификации схем данных

6.3.1 Рекомендации по использованию стандарта W3C XML SCHEMA в рамках СЭВ для описания модели данных

6.3.1.1 Специфика использования стандарта

Стандарт W3C XML Schema используется в СЭВ для описания стандартных (канонических) схем данных, регистрируемых в репозитарии метаданных СЭВ, а так же как единый стандарт описания данных интегрируемых с помощью СЭВ ИС. СЭВ не определяет дополнительных расширений стандарта и рекомендаций к применению, помимо тех, которые описаны в разделе «5.3 Каноническая модель данных и язык спецификации схем данных».

6.3.1.2 Способ организации схем данных

Схемы данных СЭВ поддерживают модульность организации схем посредством стандартных механизмов импорта, определяемых стандартом W3C XML Schema.

6.3.1.3 Средства проектирования схем данных

При проектировании схем данных СЭВ были использованы средства UML-моделирования и последующего преобразования к формату W3C XML Schema, описанные в разделе «5.3.1 Руководство по процессу разработки XML-схем».

6.3.2 Рекомендации по использованию единой описательной модели для интеграции данных и определение требований к хранилищу схем данных в рамках СЭВ

6.3.2.1 Рекомендации по использованию единого формата данных (XML Schema) для интеграции данных в СЭВ

Для возможности интеграции информационных потоков в СЭВ используется единый формат описания схем данных W3C XML Schema. Данный формат используется для описания схем двух типов:

- канонические (стандартные) схемы данных СЭВ, в рамках которых происходит взаимодействие между всеми компонентами СЭВ и интегрируемыми информационными системами;
- схемы данных конкретных интегрируемых информационных систем. Для каждой из таких, каталогизируемых средствами СЭВ, схем определяются преобразователи

соответствующих им XML-данных, к формату, определяемому каноническими схемами, и обратно.

Все передаваемые между интегрируемыми компонентами СЭВ XML-данные соответствуют одной из XML-схем, зарегистрированных в репозитарии схем данных СЭВ, что гарантирует возможность информационной интеграции систем.

Стандарт W3C XML Schema выбран для описания интегрирующих схем данных в связи со следованием СЭВ общим интегрирующим механизмам WSA, описанным в разделе «4. Вопросы интеграции в контексте Web-технологий».

СЭВ не определяет дополнительных рекомендаций по стилевому оформлению интегрирующих XML-схем, помимо тех, которые описаны в разделе «5.3.2 Руководство по стилю XML-схем».

6.3.2.2Предлагаемые требования к хранилищу схем данных

Хранилище схем данных для использования при обмене информацией между компонентами СЭВ и внешними системами представлено модулем «Репозитарий». Модуль «Репозитарий» предназначен для хранения информации о составе информационных ресурсов и используемых метаданных. В репозитарии хранится информация о структурах данных, используемых при реализации ПЭВ (Процедура Электронного Взаимодействия) и вызовах внутренних web-сервисов.

Информация о типе данных включает:

1. описание структуры данного типа (***XSD***);
2. указание типа данных, являющегося каноническим для данного типа;
3. набор инструкций для преобразования данных описываемого типа к каноническому виду и обратно;
4. наборы инструкций для полного или частичного преобразования данных описываемого типа к любому другому типу и, возможно, обратного преобразования;
5. техническое описание типа данных и его свойств на русском языке для администратора Среды (неструктурированная справочная информация – текст, графика).

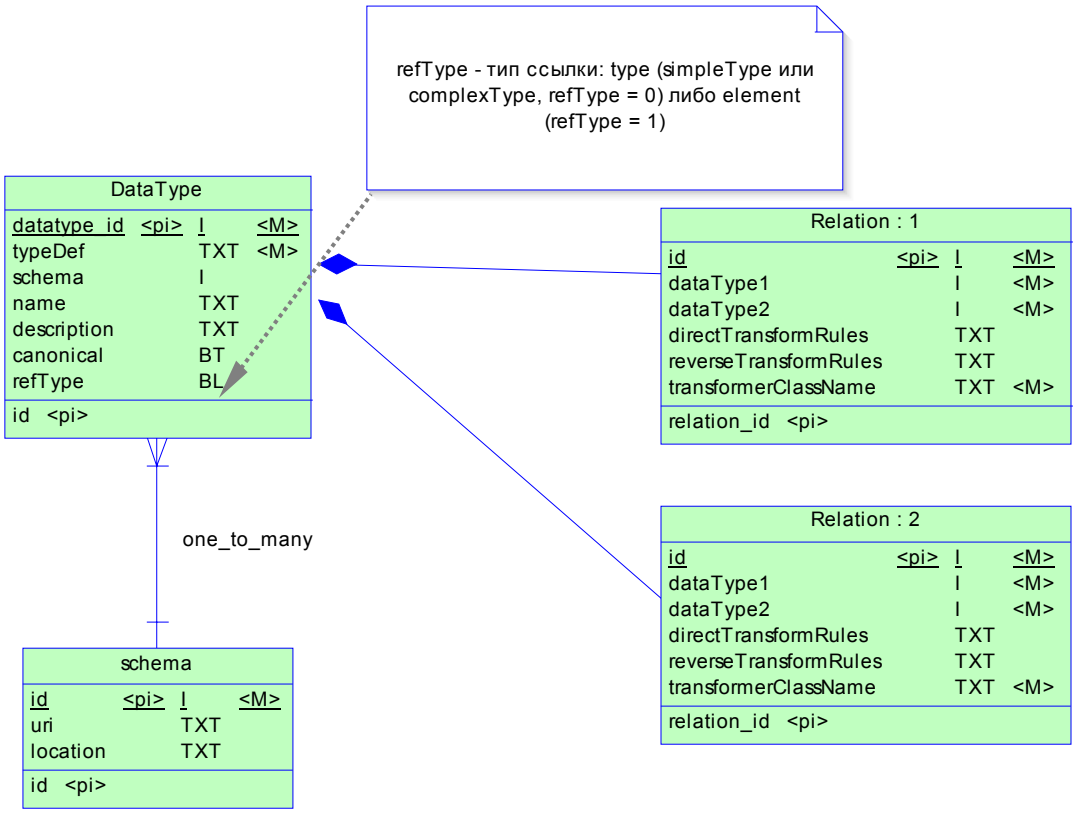


Рисунок 21. Структура данных репозитария

К основным функциям хранилища метаданных СЭВ относится:

- проверка наличия в репозитарии описания типа (осуществляется путем проверки наличия в репозитарии пространства имен, содержащего описание указанного типа);
- выборка множеств типов в соответствии со сложными критериями поиска, описывающими условия, накладываемые на значения параметров типов, принадлежность типов к ветвям ССК, принадлежность их к пространствам имен. Критерии поиска могут быть построены на основе разнообразных операций (больше, меньше, равно, вхождение подстроки, соответствие шаблону регулярного выражения и т.д.);
- автоматическое преобразование данных зарегистрированных типов в ходе функционирования Среды;
- получение основной информации о зарегистрированных типах (структура, канонический тип, инструкции преобразования к каноническому типу и обратно);
- получение инструкций о преобразовании зарегистрированных типов к любым другим типам, за исключением канонического, и обратно;

- получение справочной информации о зарегистрированных типах;
- поддержка наследственности типов данных;
- обращение к подсистеме ССК для получения описания (формата) указанного типа данных;
- описание хранящихся в системе ресурсов;
- экспорт/импорт описаний данных;
- откат инициированных действий при сбойных ситуациях и возобновление при их устранении.

6.4 Регистрация сервисов и метаинформации

6.4.1 Рекомендации по использованию стандарта OASIS UDDI

6.4.1.1 Характерные особенности реализации

Реестр сервисов (UDDI-реестр) предназначен для регистрации и обнаружения субъектов взаимодействия и предоставляемых ими услуг и представляет собой иерархическое хранилище информации о субъектах взаимодействия, их контактных лицах, осуществляемых субъектами электронных административных регламентах и технических спецификациях этих регламентов, а также о предоставляемых СЭВ процедурах электронного взаимодействия и их технических спецификациях.

Модуль UDDI предназначен для регистрации, хранения и модификации об организациях, их услугах, а так же технических спецификаций предоставляемых услуг.

В состав Oracle Application Server 10g входит реализация реестра UDDI. При этом в рамках СЭВ разработан дополнительный локальный Java API сервиса UDDI. Удаленный SOAP API должен соответствовать спецификации UDDI v2.0. Таким образом, Oracle UDDI не доступен напрямую ни другим сервисам Среды, ни сторонним организациям. На следующей диаграмме приведены компоненты модуля и их связи:

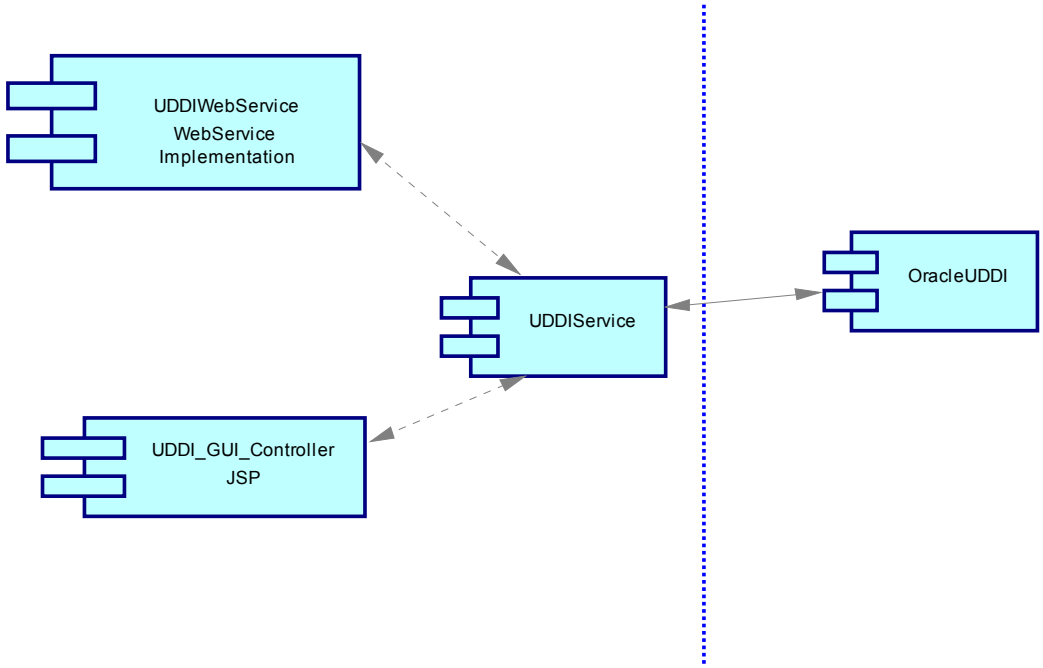


Рисунок 22. Взаимодействие компонентов модуля UDDI

6.4.1.2Соглашения по используемой модели данных

В реестре, согласно спецификации UDDI, хранятся, прежде всего, данные о предприятиях и об осуществляемых ими услугах. Для хранения указанных сведений в базе данных реестра сервисов используются основные сущности – предприятие, услуга, связывающий шаблон и тип услуги, – а также набор дополнительных, предназначенных для описания отдельных характеристик основных сущностей.

Предприятие (businessEntity). Информация о конкретном предприятии включает: системный идентификатор предприятия, набор его названий на нескольких языках, набор кратких описаний на нескольких языках, информацию о контактных лицах, список категорий и идентификаторов, характеризующих предприятие, а также адреса ресурсов, содержащих более подробную информацию о предприятии.

Услуга (businessService). Предприятию соответствует список предоставляемых им услуг. Каждый элемент списка содержит системный идентификатор услуги, набор ее описаний на нескольких языках, перечень категорий, описывающих эту услугу, ссылки к ресурсам, содержащим детальное описание услуги.

Связывающий шаблон (bindingTemplate). Ассоциированные с каждой из услуг списки связующих шаблонов, указывающих на технические спецификации типов услуг и параметры этих спецификаций применительно к конкретным услугам.

Тип услуги (tModel). Тип услуги (ее общие технические спецификации) определяется связанной с ней посредством шаблона сущностью tModel. Согласно спецификации UDDI, различные предприятия могут предлагать один и тот же тип услуг. Описание типа услуги

включает системный идентификатор, собственное имя, данные об операторе, опубликовавшем тип услуги, список категорий, описывающих его, сведения о технических спецификациях данного типа услуг, например, определения интерфейсов, форматы сообщений, протоколы сообщений, протоколы безопасности и т. п.

Кроме того, сущность tModel может задавать некоторую систему идентификации или классификации. В этом случае tModel рассматривается как пространство имен, в рамках которого задаются пары «ключ-значение» для идентификации или классификации сущностей реестра.

Помимо приведенных выше основных сущностей, предусмотренных стандартом UDDI, в рамках работ по ФЦП «Электронная Россия», реестр доработан таким образом, что учитывается информация о пользователях, осуществляющих публикацию предприятий и предоставляемых ими услуг. Для этих целей в состав базы данных реестра СЭВ включается основная сущность PUBLISHER, хранящая данные о пользователе, и связанная с ней дополнительная сущность AUTH_TOKEN, хранящая информацию о пользовательских сессиях. Для авторизации пользователей реестра выполняется обращение к подсистеме безопасности СЭВ.

Ниже приведены описания таблиц базы данных реестра UDDI и их краткая характеристика. Более подробные сведения приведены в UDDI Version 2.0 Data Structure Reference (www.uddi.org).

Таблица 4. Таблиц базы данных реестра UDDI и их краткая характеристика

№	Таблицы	Связи (по FK)	Описание хранимых данных
1	BUSINESS_ENTITY	PUBLISHER	Описание организаций, предоставляющих услуги
2	DISCOVERY_URL	BUSINESS_ENTITY	URL web-ресурсов, содержащих описания организаций
3	BUSINESS_NAME	BUSINESS_ENTITY	Названия организаций (возможно, на разных языках)

4	BUSINESS_DESCR	BUSINESS_ENTITY	Описания организаций (возможно, на разных языках)
5	CONTACTS	BUSINESS_ENTITY	Описания контактов с организациями (контактных лиц или должностей)
6	CONTACT_DESCR	CONTACTS	Описания способов обращения к контактным лицам
7	EMAIL	CONTACTS	Адреса электронной почты контактных лиц
8	PHONE	CONTACTS	Телефоны контактных лиц
9	ADDRESS	CONTACTS	Адреса контактных лиц
10	ADDRESS_LINE	ADDRESS	Значения отдельных полей адреса
11	BUSINESS_IDENTIFIER	BUSINESS_ENTITY TMODEL	Идентификаторы организаций в соответствии с системой идентификации, описываемой tмоделью
12	BUSINESS_CATEGORY	BUSINESS_ENTITY TMODEL	Классификация организаций в соответствии с системой классификации, описываемой tмоделью
13	BUSINESS_SERVICE	BUSINESS_ENTITY	Описание услуг, предоставляемых организациями
14	SERVICE_NAME	BUSINESS_SERVICE	Названия услуг (возможно, на разных языках)
15	SERVICE_DESCR	BUSINESS_SERVICE	Описания услуг (возможно, на разных языках)

16	SERVICE_CATEGORY	BUSINESS_SERVICE TMODEL	Классификация услуг организаций в соответствии с системой классификации, описываемой tмоделью
17	BINDING_TEMPLATE	BUSINESS_SERVICE	Технические спецификации услуг, предоставляемых организациями
18	BINDING_DESCR	BINDING_TEMPLATE	Описание технических спецификаций
19	TMODEL_INSTANCE_INFO	BINDING_TEMPLATE	Значения параметров технических спецификаций применительно к конкретным услугам
20	TMODEL_INSTANCE_INFO_DESCR	TMODEL_INSTANCE_INFO	Описание указанных значений параметров технических спецификаций
21	TMODEL_INSTANCE_DETAILS_DOC_DESCR	TMODEL_INSTANCE_INFO	Ссылки на документы, содержащие инструкции по использованию значений параметров спецификаций, и описания этих документов
22	INSTANCE_DETAILS_DESCR	TMODEL_INSTANCE_INFO	Инструкции по использованию значений параметров спецификации
23	PUBLISHER	–	Информация о пользователе, опубликовавшем информацию о tмодели или организации

24	PUBLISHER_ASSERTION	BUSINESS_ENTITY BUSINESS_ENTITY TMODEL	Указание наличия между организациями отношения из пространства имен, описываемого tмоделью
25	AUTH_TOKEN	PUBLISHER	Пользовательские сессии реестра
26	TMODEL	PUBLISHER	Технические спецификации обращений к услугам или пространства имен для идентификации или классификации
27	TMODEL_IDENTIFIER	TMODEL TMODEL	Идентификаторы tмоделей в соответствии с системой идентификации, описываемой второй tмоделью
28	TMODEL_CATEGORY	TMODEL TMODEL	Классификация tмоделей в соответствии с системой классификации, описываемой второй tмоделью
29	TMODEL_DESCR	TMODEL	Описания tмоделей (технических спецификаций или пространств имен)
30	TMODEL_DOC_DESCR	TMODEL	Ссылки на документы, содержащие инструкции по использованию значений параметров спецификаций, и описания этих документов

В данном проекте в ЕСД будет использован реестр UDDI, построенный на основе Oracle Application Server Enterprise Edition 10g.

6.4.1.3Описание используемых классификаторов

В UDDI реестре СЭВ будут использоваться только стандартные классификаторы UDDI. В дополнение к этому на сегодняшний день в системе содержится 13 классификаторов: ISO, NAICS, SIC, UNSPSC, а также такие как ИНН, ОКОНХ, ОКПО и другие.

6.4.1.4Механизмы доступа к данным реестра

Механизм доступа к данным реестра UDDI представлен GUI-интерфейсом пользователя реестра UDDI, интегрированным в административный портал СЭВ портал. Исполнительная среда клиентской части пользовательского интерфейса – сервер Tomcat 5.

Интерфейс пользователя предоставляет возможность простого и расширенного поиска организаций и услуг, интерфейс для публикации данных об организациях и услугах, и выборку данных, опубликованных пользователем. При этом интерфейс выбора ССК (выбор классификатора и значений классификатора) интегрирован в экраны пользовательского интерфейса. При просмотре результатов поиска предоставляется регистрационная информация об услугах и организациях, а так же, в случае, если услуга является ПЭВ, ссылку на страницу запуска ПЭВ.

6.4.1.5Механизмы публикации данных в реестре

Механизм публикации данных в реестре UDDI представлен GUI-интерфейсом администратора, интегрированными в административный портал СЭВ портал. Исполнительная среда административной части пользовательского интерфейса – сервер Tomcat 5.

Интерфейс администратора предоставляет возможность добавления, удаления и изменения опубликованных данных, доступ к настройке пользовательского интерфейса и настройкам соединения с Oracle UDDI (имя, пароль, *url* и т.п.).

6.4.1.6Роль UDDI реестра в общей инфраструктуре

Регламент взаимодействия реестра UDDI с другими подсистемами СЭВ представлен следующими требованиями к его функциональности:

- необходимость проверять наличие типов данных, заявленных в публикуемом сервисе (путем проверки передаваемого в виде URL-адреса WSDL-описания) в репозитарии Среды, в случае отсутствия описания нужно отказать в регистрации сервиса;

- необходимость выполнять авторизацию пользователей, используя API сервиса авторизации;
- необходимость выполнять данные об организации в подсистему ССК при публикации новой организации;
- необходимость проверять наличие указанных при публикации классификаций в ССК и при их отсутствии отказать в публикации.

Модуль UDDI предоставляет следующие интерфейсы для других подсистем: *IUDDIService* и *UDDIInterceptor*. Структура этих интерфейсов приведена на следующей диаграмме:

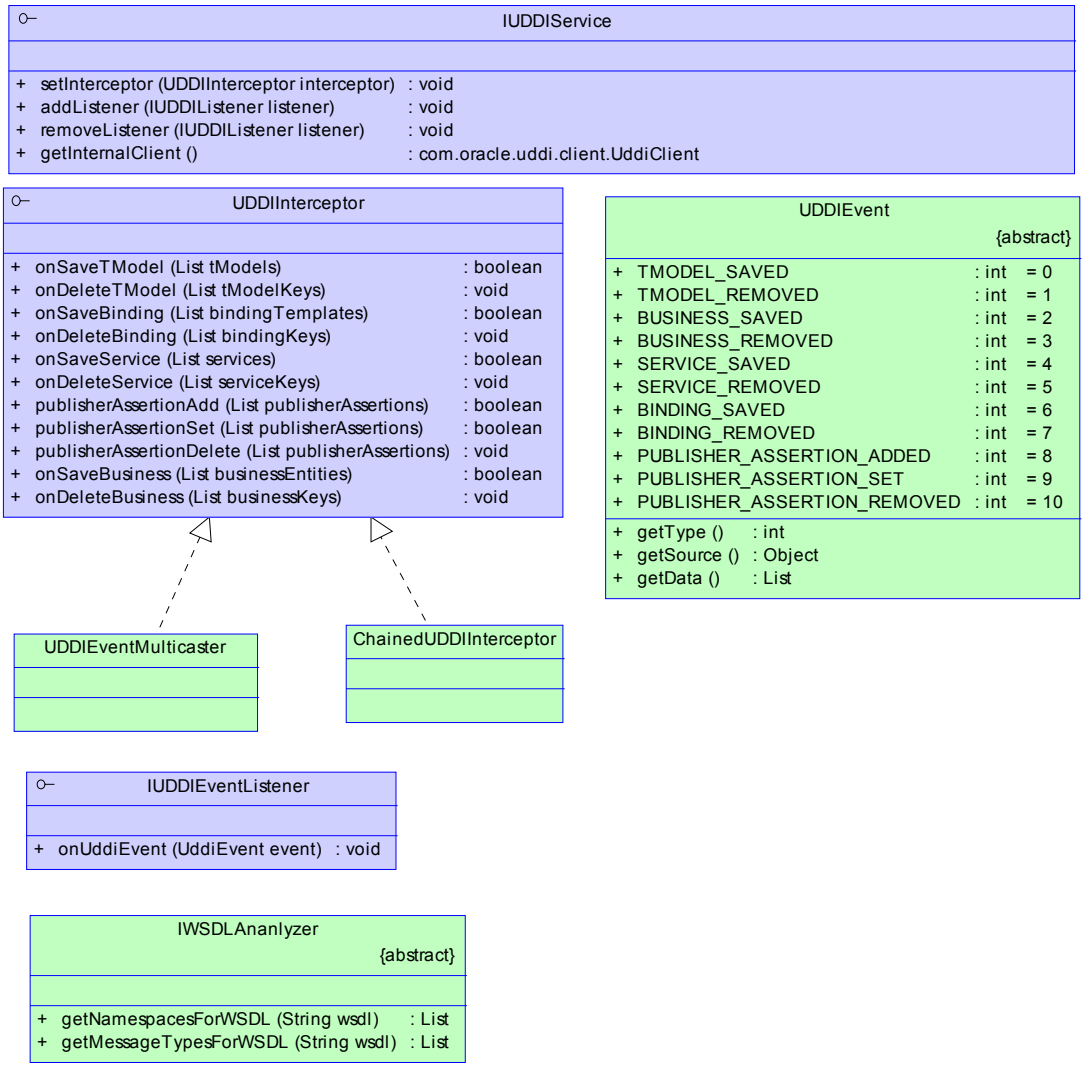


Рисунок 23. Программные интерфейсы компонентов модуля UDDI

Следующая диаграмма описывает схему взаимодействия *UDDIService* с подсистемой репозитария и подсистемой классификаторов (ССК):

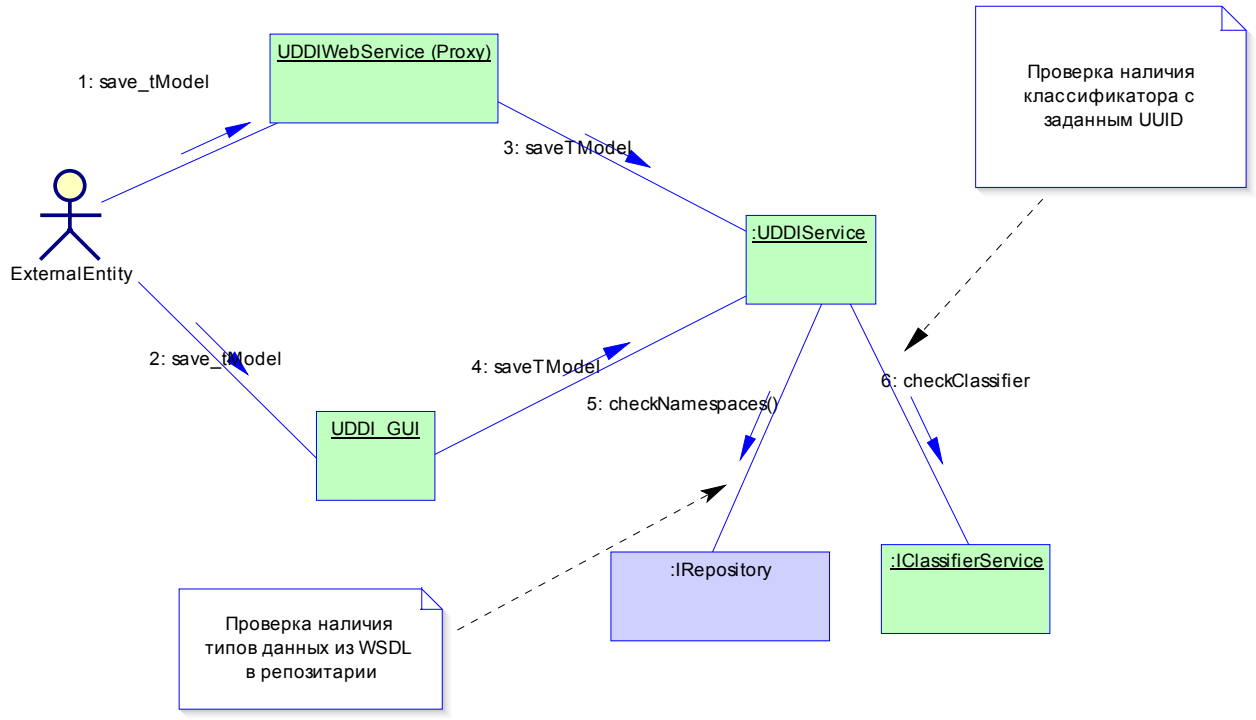


Рисунок 24. Взаимодействие модуля UDDI с модулем ССК

6.5 Безопасность

Задачи безопасности в СЭВ в рамках работ данного года предполагается решать через два основных механизма: авторизационный сервис СЭВ и протокол HTTPS для транспорта в сети Интернет. Протокол HTTPS был выбран как наиболее простой и надежный механизм, который на данный момент широко используется во многих системах. WS-I Basic Profile 1.0 рекомендует использование HTTPS для задач безопасности. Все остальные стандарты (WS-Security, WS-Trust, WS-SecureConversation и т.д.) представляются интересными для дальнейших работ, но у них есть несколько недостатков:

- они не являются общепризнанными стандартами, пока это инициатива группы компаний;
- они не используются на данный момент повсеместно, а это означает, что, они еще далеки от совершенства;
- реализация данных стандартов будет значительно более трудоемка, чем реализация HTTPS.

6.6 Транзакции

6.6.1Обоснование выбора модели бизнес-транзакций

Для реализации транзакционности в СЭВ была выбрана модель бизнес-транзакций (LRT). Язык BPEL4WS описания маршрута ПЭВ поддерживает модель бизнес-транзакций.

Атомарные транзакции являются следующим уровнем детализации и их поддержка должна быть реализована не в СЭВ, как в интеграционном механизме, а в самих сторонних Web-сервисах. Естественно для обеспечения данной возможности процесс должен предавать необходимую информацию Web-сервисам. Передача данной информации будет обеспечена специальными требованиями, которые предъявляются к Web-сервисам, а также к самим ПЭВ.

6.6.2Рекомендации по реализации модели бизнес-транзакций

Реализация модели бизнес-транзакций средствами языка BPEL4WS полностью соответствует описанному в разделе «5.6.2 Рекомендации по реализации координирующих протоколов бизнес-транзакций на базе средств обработки исключительных ситуаций и компенсации языка описания АРП BPEL4WS» механизму декларации долгоживущих транзакций.

Для возможности реализации атомарных транзакций на маршруте ПЭВ на взаимодействующие сервисы накладывается ряд ограничений по оформлению их внешних интерфейсов.

Для большинства сервисов в любой момент до завершения выполнения web-сервиса в рамках некоторого экземпляра ПЭВ может быть произведен откат выполнения сервиса, т.е. сервис в рамках ПЭВ может быть приведен в исходное состояние.

Разрабатываемый субъектом web-сервис должен обладать методом, позволяющим произвести откат выполнения сервиса в рамках некоторого экземпляра ПЭВ к его первоначальному состоянию. Единственным параметром входного сообщения этого метода является идентификатор обратившегося к сервису экземпляра ПЭВ; единственным параметром выходного – результат попытки отката сервиса (откат сервиса произведен, откат сервиса не может быть произведен, сервис уже находится в процессе отката, обращение указанного экземпляра ПЭВ уже завершено и поэтому откат невозможен, обращение указанного экземпляра ПЭВ не происходило). Сервисы, откат которых непредусмотрен, должны обладать указанным методом, но всегда возвращать результат «откат сервиса не может быть произведен».

```
<message name="RollbackRequest">
      <part name="eip_id" type="string"/>
</message>

<message name="RollbackResponse">
      <part name="result" type="string"/>
</message>

<operation name="RollbackWS">
<input message="RollbackRequest"/>
<output message="RollbackResponse"/>
</operation>
```

6.6.3Реализация службы транзакций СЭВ

В части работ 2004 года реализована служба транзакций СЭВ посредством поддержки на уровне BPEL4WS-конструкций (Fault-handler&Compensation-handler), координационный протокол реализуется для конкретного сервиса-участника ПЭВ на основе синхронных вызовов.

6.6.4Описание общей структуры контекста бизнес-транзакции, передаваемого в заголовках сообщений

В дальнейшем возможно использование стандартов WS-Coordination. WS-Coordination основная спецификация группы компаний IBM, BEA и Microsoft, описывающая механизмы координации Web-сервисных операций. Другие спецификации этой группы компаний, в частности WS Transaction, WA Atomic Transaction и WS Business Activity Framework, опираются на нее и расширяют ее, распространяя на более узкие области управления атомарными и бизнес-транзакциями. В среде, соответствующей требованиям WS-Coordination, согласование достигается за счет передачи блока XML-данных с информацией о контексте транзакции. Спецификация описывает теги, используемые для описания этого блока, требования к программным средствам его передачи и требования к реализациям протоколов координации действий участников (точное описание последних — задача других спецификаций вроде WS-Transaction). Благодаря этим механизмам WS-Coordination позволяет существующим системам обработки транзакций, workflow и другим координаторам “обернуть” в стандартизованную форму используемые ими приватные протоколы и, как следствие, получить возможность для работы в гетерогенной среде.

6.7 Рабочие процессы

6.7.1 Рекомендации по использованию стандарта описания рабочих процессов BPEL4WS

6.7.1.1 Обоснование выбора стандарта и поддерживаемой версии стандарта

Выбор стандарта BPEL4WS 1.1 для описания маршрута ПЭВ в рамках СЭВ обусловлен анализом языков спецификации потоков работ, приведенном в разделе «4.7 Рабочие процессы», а так же тем, что данная версия стандарта проходит процесс непосредственной стандартизации в рамках консорциума OASIS.

6.7.1.2 Описание технологической платформы интерпретатора BPEL4WS-процессов

Модуль Среды Электронного Взаимодействия (СЭВ) BPEL4WS обеспечивает размещение описаний Процедур Электронного Взаимодействия (ПЭВ), запуск, исполнение и мониторинг хода выполнения ПЭВ.

Также, в целях проверки соответствия сценариев и интерфейсов ПЭВ предъявляемым к ним требованиям модуль исполнения ПЭВ обеспечивает:

- поддержку определения контрольных точек на маршруте ПЭВ в ходе разработки XML-описаний ПЭВ;
- средства автоматической проверки корректности WSDL-описания ПЭВ в качестве web-сервиса при его регистрации в реестре UDDI Среды;
- средства автоматической проверки соответствия WSDL-описаний формальным требованиям, предъявляемым к описанию процедур электронного взаимодействия;
- средства автоматической проверки соответствия сценариев ПЭВ требованиям, связанным с контролем последовательности операций по управлению сессиями;
- средства автоматической проверки соответствия сценариев ПЭВ требованиям, определяющим последовательность команд по управлению транзакциями и ограничения на их применение;
- средства автоматической проверки соответствия сценариев ПЭВ требованиям, касающимся последовательности обращений к методам инициализации web-сервисов и завершения их работы.

- Основные компоненты модуля:
- сервис ядра СЭВ – *BPELService*, обеспечивает программный интерфейс к BPEL-PM;
- сервис исполнения;
- Web-сервис *BPELProxy*, обеспечивающий передачу сообщений извне;
- интерфейс пользователя;
- интерфейс администратора.

6.7.1.3Описание подмножества используемых конструкций BPEL4WS

Для описания маршрута ПЭВ может использоваться полный набор конструкций, предлагаемый языком BPEL4WS.

6.7.1.4Описание сделанных расширений стандарта

В рамках работ 2004-ого года расширение схемы BPEL4WS не предусматривается, т.е. бизнес-процесс ПЭВ будет описываться только с помощью BPEL4WS документа Все необходимые конструкции на этапе развития СЭВ в 2004г. (бизнес-транзакции, параллельное выполнение, условия, таймеры и т.д.) содержатся в языке BPEL4WS. Все дополнительные конструкции предусматривается реализовывать через вызовы специализированных Web-сервисов, что не противоречит стандарту.

6.7.1.5Описание системных переменных, присутствующие в описании ПЭВ, и правил оформления вызовов системных WEB-сервисов

Для решения задач отладки и исполнения ПЭВ среде административных сервисов определены сервис исполнения и сервис отладки ПЭВ. Вызов операций данных сервисов на маршруте ПЭВ оформляется в соответствии со следующими дополнительными требованиями к именованию системных вызовов.

6.7.1.5.1 Системные пространства имен

Для поддержки вызовов сервиса исполнения и отладки в ПЭВ необходимо вводить два системных пространства имен:

- **urn:breakpointService** – пространство имен сервиса отладки;
- **urn:PEINameSpace** – пространство имен сервиса исполнения.

6.7.1.5.2 Системные связи участников

Для поддержки вызовов сервиса исполнения и отладки в ПЭВ необходимо вводить две системные связи участников (элементы **<partnerLink>**):

- связь с сервисом исполнения:

<**partnerLink** **name**="Hidden.executionServicePL"
partnerLinkType="tns:executionLT"
myRole="process"
partnerRole="executionController"/>

- связь с сервисом отладки:

<**partnerLink** **name**=" Hidden.breakServicePL"
partnerLinkType=" tns:breakLT"
myRole="process"
partnerRole="executionController"/>

6.7.1.5.3 Декларация вызовов операций сервиса исполнения ПЭВ

Для возможности вызова операций сервиса исполнения ПЭВ каждое описание ПЭВ должно включать набор системных переменных, определяющих входящие и исходящие сообщения операций сервиса исполнения. Сам синтаксис вызова операций сервиса должен соответствовать строгим соглашениям по именованию. Это подразумевает использование стандартных префиксов в именах элементов BPEL4WS **<invoke>**, моделирующих данные вызовы. Следующие разделы для каждой вызываемой операции сервиса исполнения содержат описание входных и выходных переменных (элементы BPEL4WS <variable>) и самого вызова (элемент BPEL4WS <invoke>).

6.7.1.5.3.1 Операция waitForUser

6.7.1.5.3.1.1 Сигнатура и назначение операции

waitForUser(String peiSessionId, String url) throws RemoteException – сообщает об ожидании ввода информации от пользователя (передается идентификатор сессии и ссылка на форму ввода данных).

6.7.1.5.3.1.2 Входная переменная

<**variable** **name**="Hidden.waitForUserRequest"
messageType="exec:waitForUserRequest"/>

6.7.1.5.3.1.3 Выходная переменная

<**variable** **name**="System.waitForUserResponse"
messageType="exec:waitForUserResponse"/>

6.7.1.5.3.1.4 Декларация вызова

<**invoke name**="**WaitForUser.**Name"
suppressJoinFailure="no"

partnerLink="Hidden.executionServicePL"
portType="exec:PEIExecutionService"
operation="waitForUser"
inputVariable="Hidden.waitForUserRequest"
outputVariable="System.waitForUserResponse">

6.7.1.5.3.2 Операция createInstance

6.7.1.5.3.2.1 Сигнатура и назначение операции

public ResultID createInstance(String userSessionId, int classId) throws RemoteException – сообщает о запуске экземпляра ПЭВ, получает идентификатор сессии ПЭВ.

6.7.1.5.3.2.2 Входная переменная

<**variable** **name**="Hidden.createInstanceRequest"
messageType="exec:createInstanceRequest"/>

6.7.1.5.3.2.3 Выходная переменная

<**variable** **name**="System.createInstanceResponse"
messageType="exec:createInstanceResponse"/>

6.7.1.5.3.2.4 Декларация вызова

<**invoke name**="**CreateInstance.** Name"
suppressJoinFailure="no"
partnerLink="Hidden.executionServicePL"
portType="exec:PEIExecutionService"
operation="waitForUser"
inputVariable="Hidden.createInstanceRequest"
outputVariable="System.createInstanceResponse">

6.7.1.5.3.3 Операция log

6.7.1.5.3.3.1 Сигнатура и назначение операции

public Result log(String peiSessionId, String logpointName) throws RemoteException – осуществляет логирование.

6.7.1.5.3.3.2 Входная переменная

<**variable** **name**="Hidden.logRequest"
messageType="exec:logRequest"/>

6.7.1.5.3.3.3 Выходная переменная

<**variable** **name**="Hidden.logResponse"
messageType="exec:logResponse"/>

6.7.1.5.3.3.4 Декларация вызова

<**invoke name**="**Logpoint.**Name"
suppressJoinFailure="no"
partnerLink="Hidden.executionServicePL"
portType="exec:PEIExecutionService"
operation="log"
inputVariable="Hidden.logRequest"
outputVariable="Hidden.logResponse">

6.7.1.5.3.4 Операция businessLog

6.7.1.5.3.4.1 Сигнатура и назначение операции

public Result businessLog(String peiSessionId, String logpointName, String value) throws RemoteException – осуществляет логирование с указанием строки лога.

6.7.1.5.3.4.2 Входная переменная

<**variable** **name**="Hidden.businessLogRequest"
messageType="exec:businessLogRequest"/>

6.7.1.5.3.4.3 Выходная переменная

<**variable** **name**="System.businessLogResponse"
messageType="exec:businessLogResponse"/>

6.7.1.5.3.4.4 Декларация вызова

<**invoke name**="**LogpointBusiness.**Name"
suppressJoinFailure="no"
partnerLink="Hidden.executionServicePL"
portType="exec:PEIExecutionService"
operation="businessLog"
inputVariable="Hidden.businessLogRequest"
outputVariable="System.businessLogResponse">

6.7.1.5.3.5 Операция logout

6.7.1.5.3.5.1 Сигнатура и назначение операции

public Result logout(String peiSessionId) throws RemoteException – сообщение о завершении сессии ПЭВ.

6.7.1.5.3.5.2 Входная переменная

<**variable** **name**="Hidden.logoutRequest"
messageType="exec:logoutRequest"/>

6.7.1.5.3.5.3 Выходная переменная

<**variable** **name**="System.logoutResponse"
messageType="exec:logoutResponse"/>

6.7.1.5.3.5.4 Декларация вызова

<**invoke name**="**Logout.**Name"
suppressJoinFailure="no"
partnerLink="Hidden.executionServicePL"
portType="exec:PEIExecutionService"
operation="logout"
inputVariable="Hidden.logoutRequest"
outputVariable="System.logoutResponse">

6.7.1.5.4 Декларация вызовов операций сервиса отладки ПЭВ

Аналогичным образом определяется синтаксис оформления вызовов сервиса отладки.

6.7.1.5.4.1 Операция callBreakpoint

6.7.1.5.4.1.1 Сигнатура и назначение операции

callBreakpoint*(String peiSessionID)* – при вызове этой операции ПЭВ ждет ответа до тех пор, пока не будет нажата кнопка продолжения (элемент GUI средств отладки)..

6.7.1.5.4.1.2 Входная переменная

<**variable** **name**="Hidden.callBreakpointRequest"
messageType="exec:callBreakpointRequest"/>

6.7.1.5.4.1.3 Выходная переменная

<**variable** **name**="System.callBreakpointResponse"
messageType="break:callBreakpointResponse"/>

6.7.1.5.4.1.4 Декларация вызова

<**invoke name**="**Breakpoint.**Name"
suppressJoinFailure="no"
partnerLink="Hidden.breakServicePL" **portType**="exec:breakService"
operation="callBreakpoint"

inputVariable="Hidden.callBreakpointRequest"
outputVariable="break:callBreakpointResponse">

6.7.1.6 Протокол внешнего взаимодействия с ПЭВ

Запуск нового экземпляра ПЭВ всегда инициализируется синхронным сообщением:

```
<message name="peiStartInfo">
    <part name="SessionId" type="xsd:string"/>
    <part name="ClassId" type="xsd:long"/>
</message>
```

На данное сообщение запущенный экземпляр процесса должен вернуть сообщение с идентификатором сессии ПЭВ:

```
<message name="psid">
     <part name="peiSessionId" type="xsd:string"/>
</message>
```

После запуска ПЭВ, он производит обращение к операции **createInstance** сервиса исполнения для создания нового экземпляра ПЭВ в репозитарии СЭВ.

По окончании работы ПЭВ необходимо закрыть сессию пользователя вызовом операции **logout** сервиса исполнения.

6.7.1.7 Общие рекомендации по оформлению BPEL4WS-маршрута ПЭВ

На форму описания ПЭВ на языке BPEL4WS не накладывается каких-либо дополнительных ограничений помимо тех, которые обозначены в рамках стандарта.

6.8 Глобальная идентификация

На данном этапе работ 2004 года механизмы глобально-уникальной идентификации в СЭВ не используются.

В дальнейшем для глобально-уникальной идентификации в СЭВ планируется использовать RDFS – язык описания словарей RDF-терминов (классов и свойств Web-ресурсов). В отличие от XML-схем, которые описывают структуру XML-документов, ограничивают их содержание, RDF-схемы позволяют определять семантику данных, основанных на XML представлении. RDFS служит фундаментом для более богатых языков описания моделей предметных областей (языков описания онтологий), которые позволяют адаптировать к Web системы математической логики и обеспечить семантическую обработку данных.

6.9 Глоссарии

Перечень тематических глоссариев:

- Глоссарий СЭВ – пространство понятий Среды Электронного Взаимодействия.

6.9.1Глоссарий СЭВ

Обозначение	Описание
АП	Административный портал Портал, предназначенный для администрирования СЭВ и обеспечивающий возможность настройки её динамической части сообразно потребностям пользователей
АР	Административный регламент Комплексное описание порядка выполнения отдельного делового процесса, обеспечивающего реализацию функций субъекта и выполняемого им на основании собственной нормативно-правовой базы
АДМИНИСТРАТОР СЭВ	Лицо, осуществляющее конфигурирование СЭВ, формирование информационных массивов и управление основными сервисами СЭВ
АИС	Автоматизированная информационная система Система субъекта взаимодействия, состоящая из персонала и комплекса программно-технических средств и информационных массивов, предназначенного для сбора, хранения, поиска и выдачи информации потребителям по их запросам
БД	База данных Поименованная, целостная, единая система данных, организованная по определенным правилам, которые предусматривают общие принципы описания, хранения и обработки данных
ИС	Информационная совместимость Под информационной совместимостью СЭВ с другими автоматизированными системами понимается информационное взаимодействие на уровне экспорта-импорта XML-документов

Обозначение	Описание
КОНЦЕПЦИЯ WEB-СЕРВИСОВ	Концепция web-сервисов заключается в объединении и интеграции разнородных систем на основе открытых стандартов
МЕТАДАННЫЕ	Данные, которые являются описанием других данных, их структуры, содержания, ключей, индексов, местонахождения, способов использования и тому подобное
МОДУЛЬ ИСПОЛНЕНИЯ ПЭВ	Компонент ядра СЭВ, предназначенный для управления (размещения, удаления) описаниями ПЭВ, запуска, исполнения и мониторинга хода выполнения экземпляров ПЭВ
МОДУЛЬ КОНСТРУИРОВАНИЯ ПЭВ	Компонент ядра СЭВ, обеспечивающий ее администраторам возможность визуального конструирования сценариев ПЭВ, проверки синтаксической и семантической корректности этих сценариев, а также их размещения в СЭВ
МОДУЛЬ ОТЛАДКИ И ТЕСТИРОВАНИЯ ПЭВ И WEB-СЕРВИСОВ	Компонент ядра СЭВ, предназначенный для автоматической проверки корректности WSDL-описаний web-сервисов, публикуемых в реестре UDDI СЭВ, и совместимости этих web-сервисов с СЭВ, а также для автоматизированной отладки сценариев размещаемых в ней ПЭВ
ОПО	Общесистемное программное обеспечение Программы, обеспечивающие возможность выполнения ЭВМ основных функций, практически не зависящих от специфики конкретных задач и областей применения ЭВМ
ОБЪЕКТЫ ВЗАИМОДЕЙСТВИЯ	Регламентированные информационные процессы, возникающие при реализации процедур взаимодействия между субъектами взаимодействия

Обозначение	Описание
ПОДСИСТЕМА БЕЗОПАСНОСТИ	Компонент ядра СЭВ, предназначенный для централизованного хранения идентификационной информации пользователей, авторизации сессий пользователей и экземпляров ПЭВ, а также протоколирования любых действиях, выполняемых модулями, сервисами и объектами взаимодействия в процессе функционирования СЭВ, и их результатов
ПОДСИСТЕМА ОПОВЕЩЕНИЙ	Компонент ядра СЭВ, предназначенный для обеспечения возможности подписки пользователей на сервисы и ветви классификаторов подсистемы ССК, а также для передачи сообщений внутри СЭВ между сервисами и пользователями
ПОДСИСТЕМА ССК	Компонент ядра СЭВ, предназначенный для хранения и контроля целостности используемых при организации взаимодействия в рамках СЭВ справочников, словарей и классификаторов
ПП	Пользовательский портал Портал, предназначенный для работы пользователей с СЭВ от имени субъектов взаимодействия
ПОЛЬЗОВАТЕЛЬ СЭВ	Субъект взаимодействия, зарегистрированный в СЭВ и имеющий доступ к СЭВ с целью обмена или получения информации
ПРАВОВОЕ ОБЕСПЕЧЕНИЕ СЭВ	Правовое обеспечение СЭВ включает набор нормативно-правовых документов определяющих юридический статус СЭВ, порядок ее функционирования и подключения других систем, а также регулирующих взаимодействие сторон
ППО	Прикладное программное обеспечение Программы, обеспечивающие пользователю возможность выполнять свои функции с использованием ЭВМ

Обозначение	Описание
ПЭВ	Процедура электронного взаимодействия Сценарий вызова внутренних web-сервисов объектов взаимодействия, формируемый администратором СЭВ на основании действующей нормативно-правовой базы
РЕЕСТР СЕРВИСОВ	Компонент ядра СЭВ, предназначенный для регистрации, изменения и обнаружения субъектов взаимодействия, предоставляемых ими сервисов и технических спецификаций сервисов
РЕПОЗИТАРИЙ СЭВ	Компонент ядра СЭВ, предназначенный для хранения информации о структурах данных, используемых при реализации ПЭВ и вызовах внутренних web-сервисов, используемых метаданных и сведений о составе информационных ресурсов
СУБД	Система управления базами данных Совокупность программных и языковых средств, предназначенных для управления данными в базе данных, ведения базы данных, обеспечения многопользовательского доступа к данным
СУБЪЕКТЫ ВЗАИМОДЕЙСТВИЯ	Участники регламентированных информационных процессов, возникающих при их взаимодействии посредством СЭВ
СЭВ	Среда электронного взаимодействия Система, представляющая собой инструментарий для интеграции существующих средств автоматизации государственных учреждений в единое информационное пространство в целях организации взаимодействия органов государственной власти с хозяйствующими субъектами на базе современных информационных технологий

Обозначение	Описание
ТП	Транспортные протоколы СЭВ поддерживает прием-передачу данных по протоколам HTTP, SMTP, FTP, обеспечивает коммуникации с Интернет по протоколам TCP/IP, а также обеспечивает вызов удаленных объектов посредством протокола SOAP
ЭАР	Электронный административный регламент Созданный и используемый с помощью информационных технологий административный регламент
ЯДРО СЭВ	Набор компонент и сервисов, реализующих основные функции СЭВ
WEB-СЕРВИС	Программная компонента, описанная при помощи языка WSDL, которая может быть динамически найдена и задействована через реестр сервисов, то есть код, доступный по протоколам HTTP, SMTP, FTP и возвращающий информацию в формате XML
WEB-ИНТЕРФЕЙС	Страница в Интернет, которая находится по определенному адресу и обеспечивает пользователю СЭВ возможность осуществлять прием-передачу данных, выполнять управляющие команды и просматривать результатов их исполнения в интерактивном режиме в реальном масштабе времени

6.10 Описание базовых схем данных для сквозных пилотных проектов СЭВ

6.10.1 Описание базовых структур данных для сквозных пилотных проектов СЭВ

На данном этапе требования к выделению базовых структур данных для сквозных пилотных проектов СЭВ не предъявляются. Однако в дальнейшем планируется наполнить репозиторий СЭВ совокупностью типовых схем данных и предоставить ряд преобразователей из форматов данных информационных систем, интегрируемых в ходе пилотных проектов, в форматы типовых схем данных и наоборот.

В пилотных проектах 2003-2004 годов при взаимодействие с web-сервисами СЭВ использовались следующие типы данных:

1. doOperation;
2. doOperationReturn;
3. POSTransferService;
4. peiSessionId;
5. getUrlReturn;
6. getBusinessContentReturn;
7. getStatusReturn;
8. initReturn;
9. ProductServiceSoapBinding;
10. Result;
11. ResultLink;
12. ResultIsReady;
13. ResultReport;
14. ResultgetBusinessContent;
15. CloseReturn;
16. CloseReturn;
17. IsReadyReturn;
18. GetDeliveryLinkReturn;
19. ResultReportReturn;
20. getBusinessContentReturn.

Типы данных пилотных проектов становятся каноническими после регистрации их в репозитарии СЭВ. Регистрация будет осуществляться администратором СЭВ.

6.10.2 Описание ключевых данных СЭВ в терминах стандартных схем данных

На данном этапе, требование необходимости описания ключевых данных СЭВ в терминах стандартных схем данных не предъявляется.

6.11 Контролируемые словари

6.11.1 Перечень используемых в СУВ словарей и справочников

На сегодняшний день в системе содержатся следующие классификаторы:

- Регионы;
- Отраслевой;
- Виды деятельности;
- Географический;
- Государственные предприятия;
- Частные предприятия;
- uddi-org:iso-ch:3166:1999;
- ntis-gov:naics:1997;
- ntis-gov:sic:1997;
- uddi-org:types;
- unspsc-org:unspsc;
- uddi.org:wsdl:types;
- Группы организаций.

7 Интеграция СЭВ с внешними ИС

Данный раздел состоит из трех частей. В первой части описывается текущая реализация интеграционных механизмов СЭВ. При этом освещаются такие вопросы, как:

- общая структура СЭВ;
- программно-техническая архитектура;
- описание назначения и интерфейсов административных сервисов СЭВ;
- требования к оформлению внутренних сервисов СЭВ, вызываемых в рамках процедур электронных взаимодействий.

Вторая часть данного раздела регламентирует набор требований к сторонним информационным системам, выполнение которых при разработке адаптера этой системы позволяет ей полноценно интегрироваться в СЭВ и участвовать в общем процессе межведомственного взаимодействия.

В последней части раздела приводится перечень этапов работ, которые необходимо провести для интеграции ИС в СЭВ. В этом разделе так же указываются требования к квалификации участвующих в процессе интеграции сотрудников и приблизительные трудозатраты.

7.1 Описание существующей реализации механизмов интеграции СЭВ с внешними системами в соответствии с разработанными правилами

7.1.1 Основные технические решения по интеграции разнородных информационных систем

СЭВ представляет собой систему спецификаций, регламентов и программных средств, обеспечивающих взаимодействие информационных систем органов государственной власти разных уровней (субъекты взаимодействия) при предоставлении информационных услуг посредством реализации процедур электронного взаимодействия субъектов, обращения к общегосударственным информационным ресурсам и информационным ресурсам субъектов.

Web-сервисы – доступные посредством описанных интерфейсов, при помощи языка WSDL - программные компоненты, которые могут быть динамически найдены и

задействованы через UDDI-реестр. В смысле такого определения web-сервис – это код, доступный по протоколу HTTP и возвращающий информацию в формате XML. Такое определение web-сервиса включает поддержку протокола SOAP для реализации возможностей вызова удаленных объектов, а также протокола UDDI для реализации возможностей поиска сервисов как на одном web-узле, так и в Internet в целом.

Объектами взаимодействия СЭВ являются информационные системы субъектов взаимодействия. Участие в СЭВ объектов взаимодействия осуществляется путем реализации собственных web-сервисов и обращения к web-сервисам других объектов взаимодействия. Объект взаимодействия может использовать несколько web-сервисов для реализации своего функционала.

Создание единого стандартизованного механизма взаимодействия информационных систем позволяет решить проблему организации совместной деятельности информационных систем, созданных в разное время различными разработчиками на основе различных платформ и технологий. Тем самым повышается эффективность взаимодействия органов власти разных уровней друг с другом и с хозяйствующими субъектами, создается единая инфраструктура доступа к услугам государственных органов власти.

В настоящее время лишь небольшое количество государственных информационных ресурсов доступно как органам власти, так и хозяйствующим субъектам. Это снижает эффективность межведомственного взаимодействия и взаимодействия государственных структур и хозяйствующих субъектов, повышает административные издержки.

Причиной этого является несоответствие используемых правил и стандартов предоставления информации. Каждый акт взаимодействия несогласованных информационных систем требует привлечения дополнительных ресурсов для обработки передаваемых данных.

СЭВ обеспечивает возможность для всех подключаемых к ней систем, вне зависимости от средств их реализации, осуществления обмена информацией на основе единых стандартов и протоколов взаимодействия.

Использование в рамках СЭВ единых правил описания и предоставления доступа к сервисам электронных систем позволяет технологически унифицировать сервисы в различных государственных системах и тем самым значительно расширять их функциональность без необходимости значительных дополнительных затрат.

Используемые при построении СЭВ открытые международные стандарты обеспечивают естественную интеграцию ведомственных ресурсов, а также интеграцию России в мировые информационные системы.

Обоснование выбора технологий WSA в качестве базового механизма интеграции разнородных информационных систем описано в разделе «4. Вопросы интеграции в контексте Web-технологий».

7.1.2Характерные особенности реализации механизма WEB-сервисов для интеграции компонентов СЭВ

7.1.2.1Бизнес-логика организации внутрисистемного и внешнего обмена данными в части СЭВ

СЭВ имеет модульную структуру с четким разделением функций между ее подсистемами. Выполнение функций СЭВ, а также доступ к содержащимся в ней данным и управляющим структурам осуществляется посредством специализированных подсистем (специализированных сервисов), обладающих как механизмами (на базе web-технологий) для взаимодействия с пользователями, так и, в необходимых случаях, специфическими API для администрирования этих подсистем. В процессе функционирования СЭВ процедуры обмена данными осуществляются при помощи сообщений в формате XML. Для взаимодействия между экземплярами СЭВ на различных уровнях предусмотрена специальная транспортная подсистема, которая обеспечивает обмен данными UDDI-реестров, данными о пользователях, описаниями ПЭВ, описаниями типов данных, используемых ПЭВ, данными справочников и классификаторов между соответствующими экземплярами следующих подсистем с помощью web-сервисов. Подсистема управления запросами содержит алгоритмы оптимального поиска информации в распределенной СЭВ и позволяет реализовать принцип экстерриториальности.

7.1.2.2Средства и методы адаптации функционирования СЭВ

Определение средств и методов адаптации функционирования СЭВ выполняются с учетом обеспечения межведомственного взаимодействия органов государственной власти федерального, регионального и муниципального уровней. Архитектура распределенной СЭВ строится по распределенной трехуровневой схеме и содержит взаимосвязанные узлы федерального, регионального и муниципального уровней, единую транспортную подсистему, единую подсистему управления запросами.

Узлы распределенной СЭВ взаимодействия федерального, регионального и муниципального уровней спроектированы на основе типового ядра, обязательными компонентами которого являются:

- реестр сервисов;

- репозитарий СЭВ;
- подсистема справочников, словарей и классификаторов;
- подсистема конструирования, отладки и исполнения процедур электронного взаимодействия;
- подсистема безопасности;
- подсистема оповещений;
- портал администрирования;
- пользовательский портал;
- транспортная подсистема.

На следующей диаграмме отображена схема функционирования узла СЭВ:

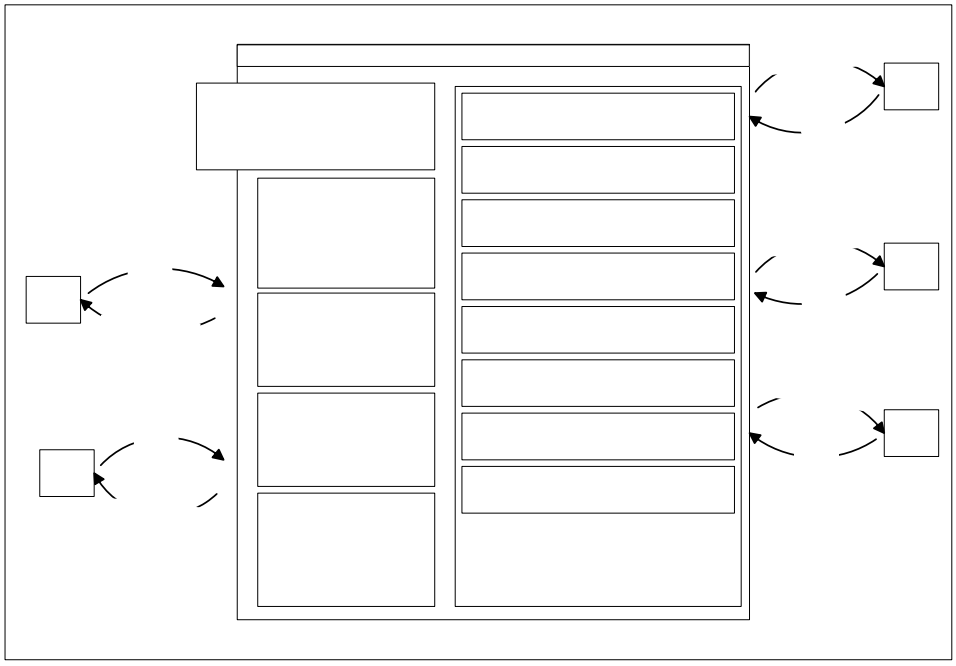


Рисунок 25. Схема функционирования узла СЭВ

Организация взаимодействия и обмена данными между различными узлами СЭВ позволит поддерживать единое информационное пространство, включающее единое адресное пространство ресурсов системы, справочников, словарей и классификаторов.

Эксплуатация узлов СЭВ федерального, регионального и муниципального уровня предполагается на большом количестве объектов различного уровня технической

оснащенности, возможностей технической поддержки и обслуживания, различных характеристик доступных каналов связи.

7.1.2.3Структура СЭВ

СЭВ включает в себя Ядро, пользовательский портал и портал администрирования. Пользовательский портал предназначен для работы пользователей с СЭВ от имени субъектов взаимодействия. Ядро СЭВ реализует все основные функции СЭВ посредством включенных в него сервисов и подсистем. Портал администрирования СЭВ обеспечивает возможность настройки динамической части СЭВ сообразно потребностям пользователей.

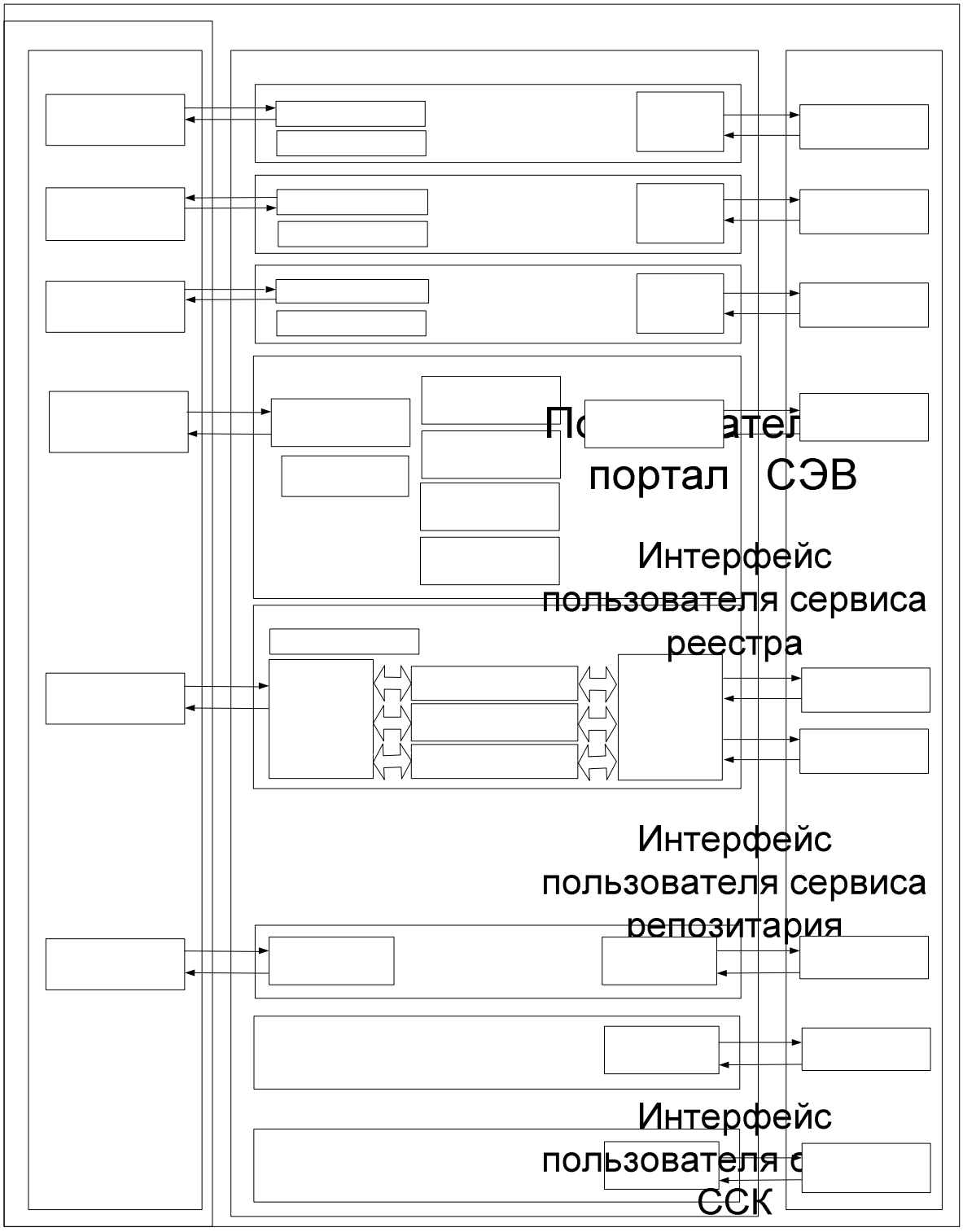


Рисунок 26. Структура СЭВ

Реестр сервисов (UDDI-реестр) предназначен для регистрации и обнаружения субъектов взаимодействия и предоставляемых ими услуг и представляет собой иерархическое хранилище информации о субъектах взаимодействия, их контактных лицах, осуществляемых субъектами электронных административных регламентах и

технических спецификациях этих регламентов, а также о предоставляемых СЭВ процедурах электронного взаимодействия и их технических спецификациях.

Репозитарий СЭВ предназначен для хранения информации о структурах данных, используемых при реализации ПЭВ и вызовах внутренних web-сервисов, используемых метаданных и сведений о составе информационных ресурсов.

Подсистема справочников, словарей и классификаторов предназначена для хранения и поддержания целостности используемых при организации взаимодействия в рамках СЭВ справочников, словарей и классификаторов.

Подсистема исполнения ПЭВ предназначена для конструирования процедур электронного взаимодействия, их отладки и тестирования на стадии разработки, хранения, а также для управления описаниями ПЭВ, их запуска, исполнения и мониторинга за ходом выполнения экземпляров ПЭВ.

Подсистема безопасности предназначена для обеспечения информационной безопасности СЭВ. Все интегрируемые с СЭВ системы должны обращаться к подсистеме безопасности для проверки прав доступа. Подсистема безопасности обеспечивает возможность регистрации пользователей и групп пользователей, хранения и изменения их регистрационных данных в унифицированной форме. Для систем, обладающих собственной системой пользователей, подсистема безопасности обеспечивает возможность определения соответствия между глобальными и локальными группами пользователей. Подсистема безопасности обеспечивает разграничение доступа к ресурсам СЭВ в процессе работы пользователей и экземпляров процедур электронного взаимодействия, а также поддержку механизма электронной цифровой подписи и инфраструктуры PKI.

Подсистема оповещений предназначена для обеспечения возможности подписки пользователей на сервисы и ветви классификаторов подсистемы ССК. Кроме этого, подсистема оповещений также отвечает за передачу сообщений внутри СЭВ между сервисами и пользователями.

Взаимодействие подсистем ядра СЭВ организовано через сервисы подсистем, а также через специализированные программные API подсистем.

Транспортная подсистема, обеспечивающая гарантированную доставку данных, поддерживает набор протоколов и стандартов, достаточный для функционирования в каналах сетей связи федеральных и региональных операторов, в том числе в защищенных сетях и специальных ведомственных сетях.

Транспортная подсистема строится с учетом требований к объему и качеству передачи информационных потоков в распределенной СЭВ. Обеспечение репликации

данных осуществляет служба репликации данных, относящаяся к транспортной подсистеме. К данной службе могут обращаться другие компоненты СЭВ для выгрузки и адресации репликационных пакетов, так же компоненты могут получать события о получении пакета для соответствующей службы.

Передачи пакетов-описателей репликационной информации основана на использовании SOAP-протокола и WSDL-описания. В этом случае обеспечивается независимость базовой логики службы репликации от транспортного уровня (HTTP, FTP, SMTP, offline-передача), и максимально используется существующая инфраструктура Web-сервисов СЭВ и сторонние библиотеки.

Для обеспечения безопасности предусматривается возможность шифрования и подписания (ЭЦП) репликационных пакетов (SSL для передачи по сети, либо DES3 для offline-передачи).

В состав транспортной подсистемы входят также подсистема управления запросами и подсистема агрегации информации. Подсистема управления запросами осуществляет поиск ресурсов в распределенной СЭВ, маршрутизацию запросов, организовывает и осуществляет, во взаимодействии с транспортной подсистемой, гарантированную доставку данных инициатору запроса. Подсистема управления запросами обеспечивает распределенную многоуровневую рассылку запросов и получение ответов. Подсистема агрегации информации, встроенная в подсистему управления запросами, обеспечивает сбор информации на узел СЭВ. При этом информация может поступать в результате исполнения запроса, так и без предварительных запросов. Для передачи информации используются транспортные механизмы службы репликации.

7.1.2.4Программно-техническая архитектура СЭВ

Программно-техническая архитектура СЭВ в реализации на платформе Sun Solaris – Oracle представлена на следующей диаграмме:

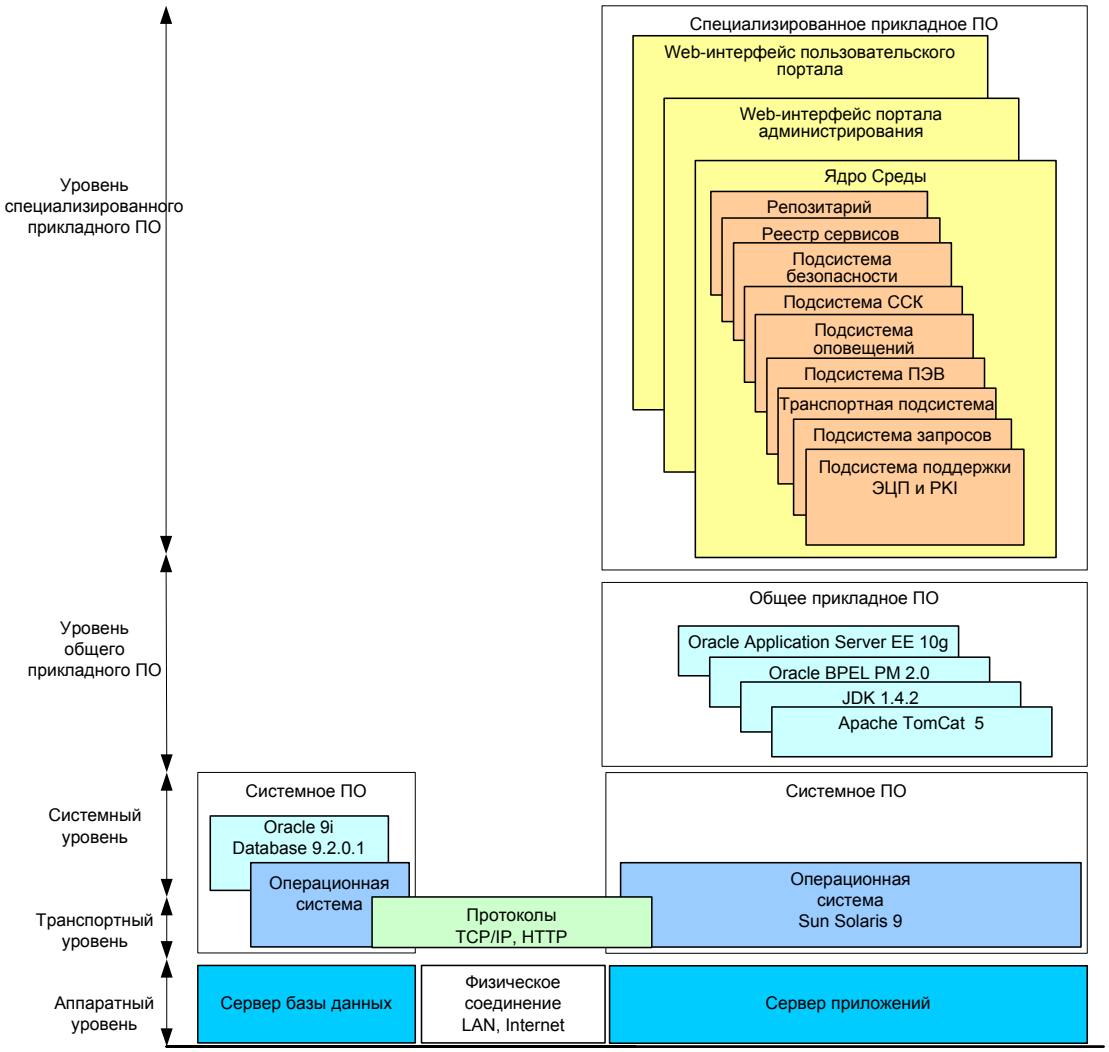


Рисунок 27. Программно-техническая архитектура СЭВ

7.1.2.4.1 Уровень системного ПО

7.1.2.4.1.1 ОС SUN Sun Solaris 9

Solaris 9 - базисная операционная система в архитектуре Sun Open Net Environment (Sun ONE), предложенной Sun для разработки, внедрения и предоставления надежных сетевых услуг. Многопоточное ядро с полной поддержкой вытесняющей многозадачности обеспечивает высокую производительность, как основных системных функций, так и приложений.

За счет автоматического использования усовершенствованной библиотеки многопоточного исполнения, существующие приложения на платформе Solaris 9 демонстрируют высокую Масштабируемость и производительность системы в условиях наиболее интенсивной пользовательской нагрузки. Файловая система UFS обеспечивает системам управления базами данных параллельный прямой ввод/вывод (Concurrent Direct I/O), что обеспечивает производительность практически на уровне физических возможностей устройств. В поставку Solaris 9 входит программное обеспечение Java 2

Platform, Standard Edition (J2SE) 1.4, что дает пользователям преимущества 64-разрядной виртуальной машины Java HotSpot. Данная функция в сочетании с оптимизатором кода Java HotSpot Server VM значительно повышает производительность сервлетов Java.

Динамическая реконфигурация позволяет изменять конфигурацию системы без перезагрузки, что сводит к минимуму время плановых и незапланированных простоев. Обновление версии Solaris можно осуществлять в процессе работы системы, что значительно сокращает обычное время прекращения обслуживания в связи со стандартным обновлением. Функция моментального снимка UFS в Solaris 9 позволяет реализовать онлайновый механизм резервного копирования посредством создания образа файловых систем на определенный момент времени.

Операционная система Solaris 9 предоставляет широкий спектр инструментальных средств администрирования, помогающих решать задачи администрирования, как системы, так и пользователей. Операционная система Solaris обеспечивает практически безопасное соединение с другими системами. Средства сетевой безопасности, реализованные в операционной системе Solaris 9, помогают обеспечить безопасную аутентификацию в сети, безопасный удаленный доступ, безопасные сетевые соединения, недоступные для прослушивания, и защиту от сетевых атак.

7.1.2.4.1.2 СУБД Oracle 9i

Ядром СУБД Oracle 9i является сервер базы данных, который поставляется в одной из четырех редакций в зависимости от масштаба информационной системы, в рамках которой предполагается его применение. При этом все варианты сервера Oracle имеют в своей основе один и тот же код и функционально идентичны, за исключением некоторых дополнительных опций, которые необходимы для специфических конфигураций (например, для поддержки кластерных архитектур необходима опция Oracle9i Real Application Clusters).

СУБД Oracle в одинаковой степени оптимизирована и для приложений оперативной обработки транзакций, и для аналитических приложений, причем их можно выполнять одновременно на одном и том же компьютере, не задумываясь о дополнительных блокировках, режимах изоляции и других технических деталях. На практике это означает, что один и тот же продукт можно с успехом использовать и как сервер оперативных баз данных, обрабатывающий интенсивный поток относительно простых и коротких транзакций, поступающих от множества пользователей, так и в качестве сервера хранилища данных, который позволяет концентрировать большие объемы данных и выполнять над ними сложные аналитические вычисления.

Одна из отличительных особенностей сервера Oracle - возможность хранения и обработки различных типов данных. Данная функциональность интегрирована в ядро СУБД и поддерживается модулем interMedia в составе Oracle Database. Он обеспечивает работу с текстовыми документами, включая различные виды поиска, в том числе контекстного; работу с графическими образами более 20-ти форматов; работу с аудио- и видео- информацией.

Основным средством доступа к базам данных Oracle из программ является (как и для других баз данных) декларативный язык запросов SQL. Этот язык по определению является платформо-независимым. На практике при разработке приложений используется процедурное расширение SQL, язык программирования PL/SQL, прототипом которому послужил язык Ада. PL/SQL - это также интерпретируемый, полностью машинно-независимый язык для разработки программ, работающих с базой данных Oracle. Фактическим стандартом для разработки стал язык программирования Java - который также полностью независим от платформы - программы на Java исполняются на всех платформах, где существует виртуальная Java-машина. В Oracle9i поддерживается и PL/SQL, и Java. То есть, в состав сервера баз данных Oracle9i включены три виртуальных машины: SQL, PL/SQL, Java.

7.1.2.4.2 Уровень общего прикладного ПО

Содержит общее прикладное ПО в составе Oracle Application Server Enterprise ed. 10g (9.0.4), Oracle BPL Proc. Manager 2.0, Apache Tomcat 4.5, JDK 1.4.2, которое обеспечивает функционирование специализированного прикладного ПО, включающего реестр сервисов, репозитарий, специализированные сервисы, пользовательский и административный порталы, включающие Web-интерфейсы.

7.1.2.4.2.1 Oracle Application Server Enterprise ed. 10g (9.0.4)

Oracle Application Server 10g - это основанная на стандартах интегрированная программная платформа, позволяющая организациям любого масштаба оперативно реагировать на меняющиеся требования рынка. Oracle Application Server 10g обеспечивает полную поддержку технологии J2EE и распределенных вычислений, включает встроенное ПО для корпоративных порталов, высокоскоростного кэширования, интеллектуального анализа бизнес-данных, быстрого развертывания приложений, интеграции бизнес-приложений, поддержки беспроводных технологий, Web-сервисов - и все это в одном продукте. Поскольку платформа Oracle Application Server 10g оптимизирована для Grid Computing, она позволяет повысить степень готовности IT-систем и снизить расходы на приобретение аппаратных средств и администрирование.

7.1.2.4.2.2 Oracle BPEL Proc. Manager 2.0

Oracle BPEL Process Manager облегчает задачу построения переносимых транзакционных и кооперационных бизнес-процессов. Программный продукт сервисно-ориентированной архитектуры BPEL Process Manager представляет собой основанную на Web-службах систему автоматизации потоков работ.

ПО позволит связывать разнородные приложения, не полагаясь при этом на специализированные механизмы интеграции. Система может функционировать на любом J2EE-совместимом сервере приложений, имеет открытые интерфейсы и базируется на WSDL и других технологиях Web-служб. BPEL Process Manager обеспечивает свободную переносимость процессов: например, компания, создавшая транзакционную систему для одного своего поставщика, может по готовым описаниям сформировать аналогичную для другого. Система поддерживает возможность использования функций Web-служб и управления бизнес-процессами, реализованных в Oracle Application Server 10g. BPEL Process Manager, имеет средства проектирования потоков работ и отладки бизнес-процессов.

Продукт создан на базе технологий, которые Oracle получила с совершенным ею недавно приобретением компании Collaxa.

7.1.2.4.2.3 Apache Tomcat 5

Apache Tomcat является высокопроизводительным сервером приложений для сервлетов (servlet) и Java Server Pages (JSP). Разрабатывается данный сервер в рамках проекта "Apache Jakarta" и свободно доступен для любого использования вместе со своими исходными текстами.

Tomcat подключается к Web-серверам Apache, Microsoft IIS и Netscape Enterprise и является платформой для текущих стандартов сервлет и JSP (Java Server Pages); он не привязан к встроенным инструментам разработки, поэтому для его развития возможно использовать любую интегрированную среду разработки или запускаемый из командной строки компилятор Java. Программам для Tomcat доступны простые средства для доступа к базам данных (вместо EJB), функционирующие как отдельные компоненты в среднем звене. Это избавляет разработчиков от необходимости составлять громоздкие программы на базе JDBC.

Tomcat способен самостоятельно обрабатывать запросы HTTP и, кроме выполнения своей основной функции - поддержи выполнения сервлетов и JSP, - может выполнять роль WEB-сервера и по обработке статической части сайта. Для сохранения конфиденциальности при обмене информацией с браузером пользователя Tomcat позволяет использовать стандарт де-факто - шифрование передаваемой/получаемой

информации по протоколу SSL/TLS (Security Socket Layer/Transport Layer Security). Наряду с самостоятельным взаимодействием с браузером пользователя, Tomcat способен работать совместно с WEB-серверами.

7.1.2.4.2.4 JDK 1.4.2

Базовый комплект разработчика приложений на Java – Java Development Kit (JDK) поддерживает ряд функций и свойств, которые позволяют разработчикам применять разнообразные подходы к построению решений, основанных на технологии Java. JDK состоит из Виртуальной машины (Java Virtual Machine), Java Core Classes, и сопутствующих файлов. Набор инструментальных средств Abstract Windowing Toolkit (AWT), поддерживающий модель делегированных событий, возможности печати и функции вырезания, копирования и переноса (cut/copy/paste), поддерживает богатый графический интерфейс пользователя (GUI) и позволяет создавать сложные приложения с использованием этого интерфейса. Поддержка программных модулей JavaBeans позволяет достичь независимости от программно-аппаратной платформы. Кроме того, функции безопасности, такие как цифровая подпись и агрегация сообщений, делают Java идеальной платформой для работы с критически важными сетевыми приложениями в масштабе предприятия.

7.1.3Общая структура интегрируемых с помощью технологии WEB-сервисов компонентов СЭВ с внешними системами

Общая структура взаимодействия компонентов СЭВ приведена на следующей диаграмме:

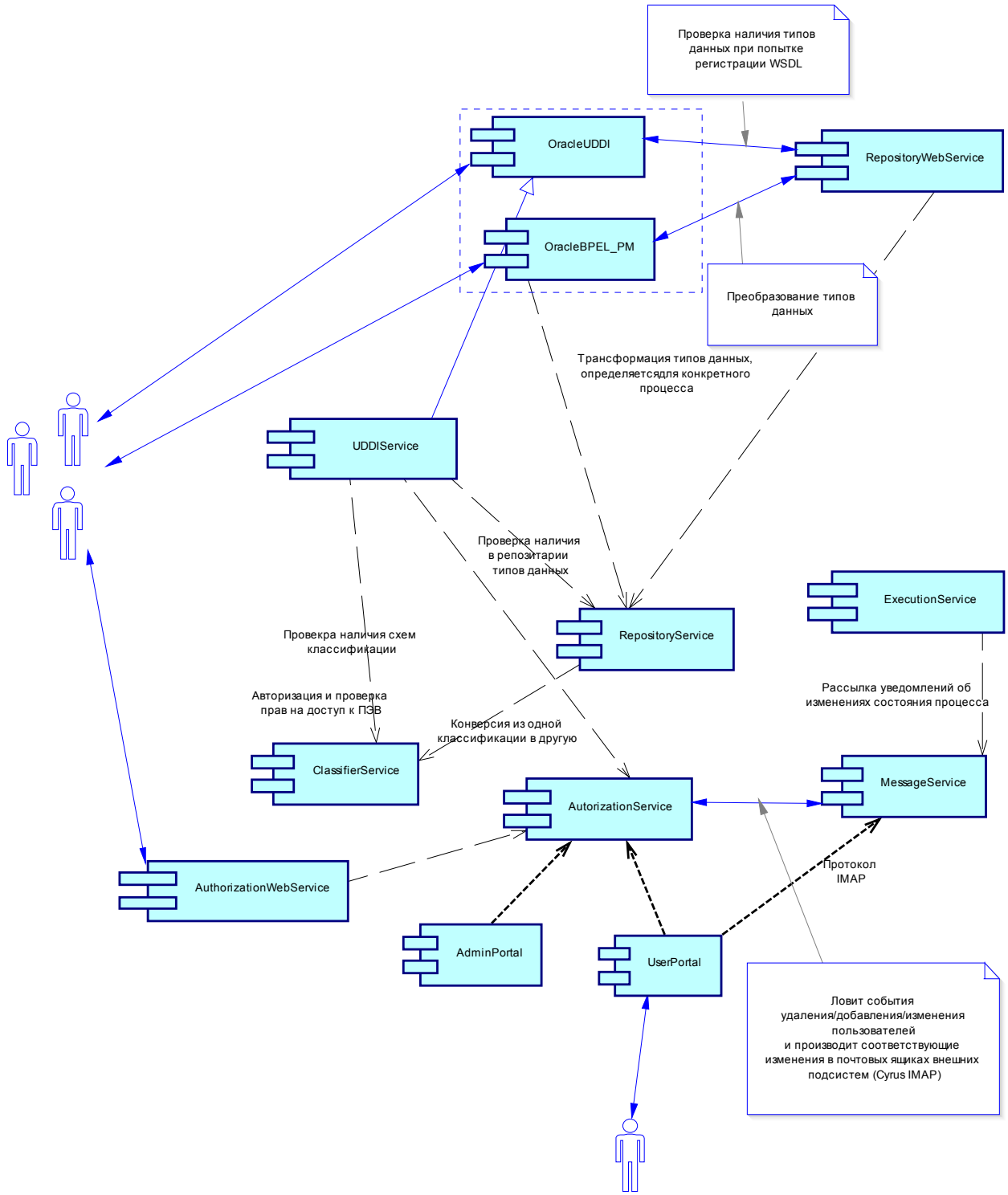


Рисунок 28. Общая схема взаимодействия компонентов СЭВ

Внешние системы при взаимодействии с СЭВ обращаются к следующим web-сервисам и компонентам СЭВ, которые, в свою очередь, могут обращаться к другим внутренним web-сервисам СЭВ для выполнения действия:

1. OracleUDDI – для регистрации в UDDI-реестре СЭВ, а также для получения информации о зарегистрированных сервисах.

2. Oracle BPEL_PM – для запуска ПЭВ или при обращении к исполняющей среде СЭВ.
3. AuthorizationWebServise – при авторизации пользователя в системе.

Внешние пользователи могут взаимодействовать с СЭВ через пользовательский портал (UserPortal), который содержит следующие пользовательские компоненты:

3. Портлет голосований (включает администраторский интерфейс и построитель).
4. Лента новостей (включает администраторский интерфейс).
5. Контекстная помощь.
6. UDDI-поиск.
7. Форма входа в систему (должна быть возможность напоминания/изменения пароля при утрате, возможность размещения заявки на регистрацию пользователя с последующим подтверждением).

Для авторизовавшихся пользователей:

1. IMAP-портлет (для зарегистрированных пользователей).
2. Список запущенных ПЭВ, с возможностью перехода к каждому экземпляру.
3. UDDI-публикация.

7.1.4Описание административных сервисов СЭВ

7.1.4.1Описание сервисов модуля авторизации

7.1.4.1.1 Общее описание

Сервис авторизации предназначен для централизованного хранения идентификационной информации и авторизации сессий пользователей и экземпляров ПЭВ.

Реализуемый функционал:

- сервис обеспечивает хранение и изменение данных о пользователях, ролях, группах и привилегиях (ролях);
- сервис обеспечивает авторизацию пользователей системы;
- сервис поддерживает систему хранения информации о сессиях пользователей и сессиях ПЭВ, обеспечивает удаление сессий по истечении срока действия;
- сервис обеспечивает получение событий системы хранения идентификационной информации о пользователях, группах и ролях (создание, удаление, изменение).

Web-интерфейс системы авторизации обеспечивает:

- управление пользователями (привязка к организациям реестра);
- управление привилегиями;
- управление ролями (привязка к сервисам реестра);
- распределение пользователей по группам (привязка к значениям классификатора «Группы пользователей СЭВ»);
- распределение ролей по группам (привязка к значениям классификаторов).

7.1.4.1.2 Программные интерфейсы

Для обеспечения взаимодействия ПЭВ и удаленных клиентов с сервисом авторизации, сервис предоставляет клиентский API (веб-сервис). Функциональность клиентского API веб-сервиса авторизации представлена следующим списком операций:

- [Result result, String sessionId] login(String username, String password) – авторизация пользователя, возвращается идентификатор сессии;
- [Result result, String peiSessionId] loginPei(String userSessionId, Long peiInstanceId) – авторизация ПЭВа;
- [String result, Long sessionType] getSessionType(String sessionId) – возвращает тип сессии (пользовательская или ПЭВа);
- Result result logout(String sessionId) – завершение сессии (для любого типа сессий);
- [Result result, Integer userId] getUserId(String sessionId) – получение id пользователя (для любого типа сессий);
- [Result result, User user] getUserInfo(String sessionId) – получение полной информации о пользователе (для любого типа сессий);
- [Result result, Role [] roles] getRolesForService(String sessionId, String serviceId) – получение списка ролей для пользователя (для любого типа сессий);
- [String result, String userId, Long peiInstanceId] getPeiSessionInfo(String peiSessionId) – получение параметров сессии ПЭВа;
- [Result result, String question] getSecretQuestion(String login) – получение секретного вопроса;

- [Result result, String password, String eMail] changePassword(String login, String question, String answer) – смена пароля.

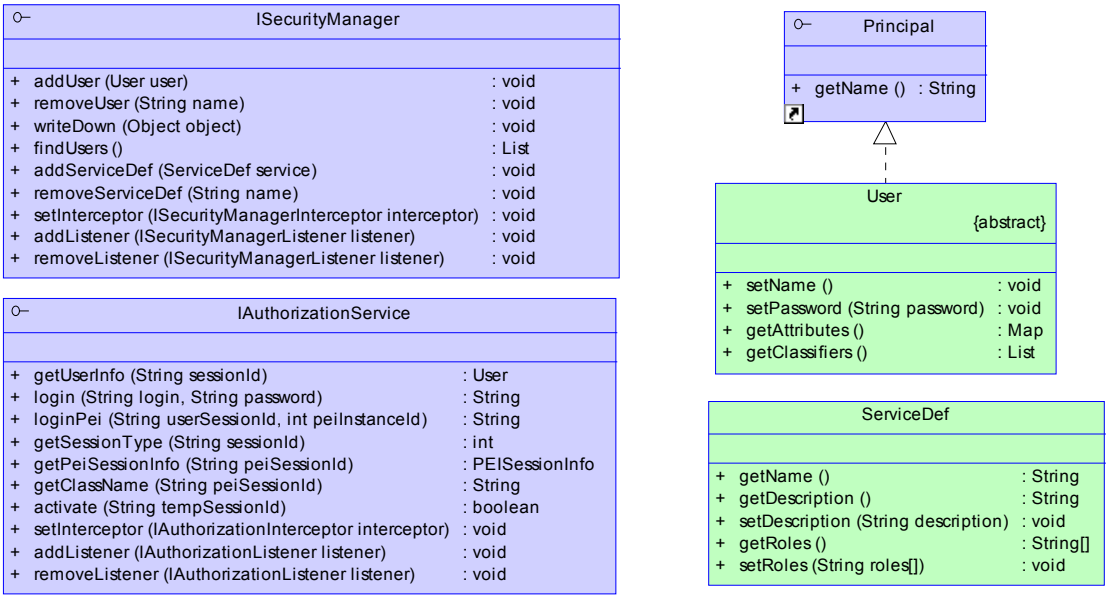
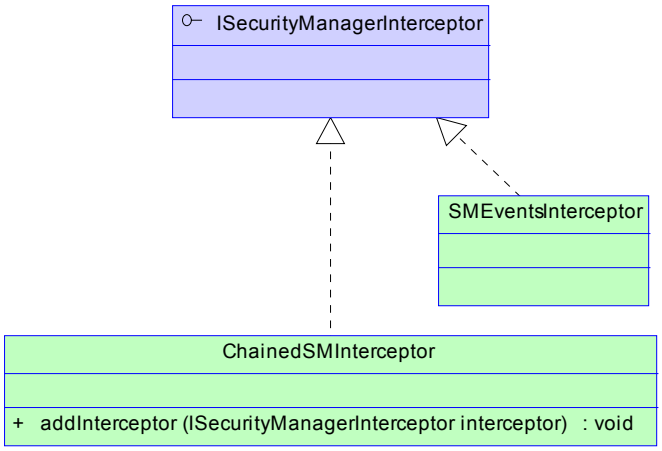


Рисунок 29. Программные интерфейсы сервисов модуля авторизации

Сервис реализует два интерфейса – *ISecurityManager* и *IAuthorizationService*. Первый отвечает за управление базой пользователей и описателей сервисов, второй – за обработку запросов на авторизацию и управление сессиями. Класс *ServiceDef* используется для внесения в базу описания сервиса (имя, строка описания, набор возможных ролей).

Сервис авторизации позволяет устанавливать «перехватчики» и слушатели событий входа/выхода и управления пользователями. Передача событий слушателям не должна влиять на ход исполнения процедуры (не реагировать на исключения).



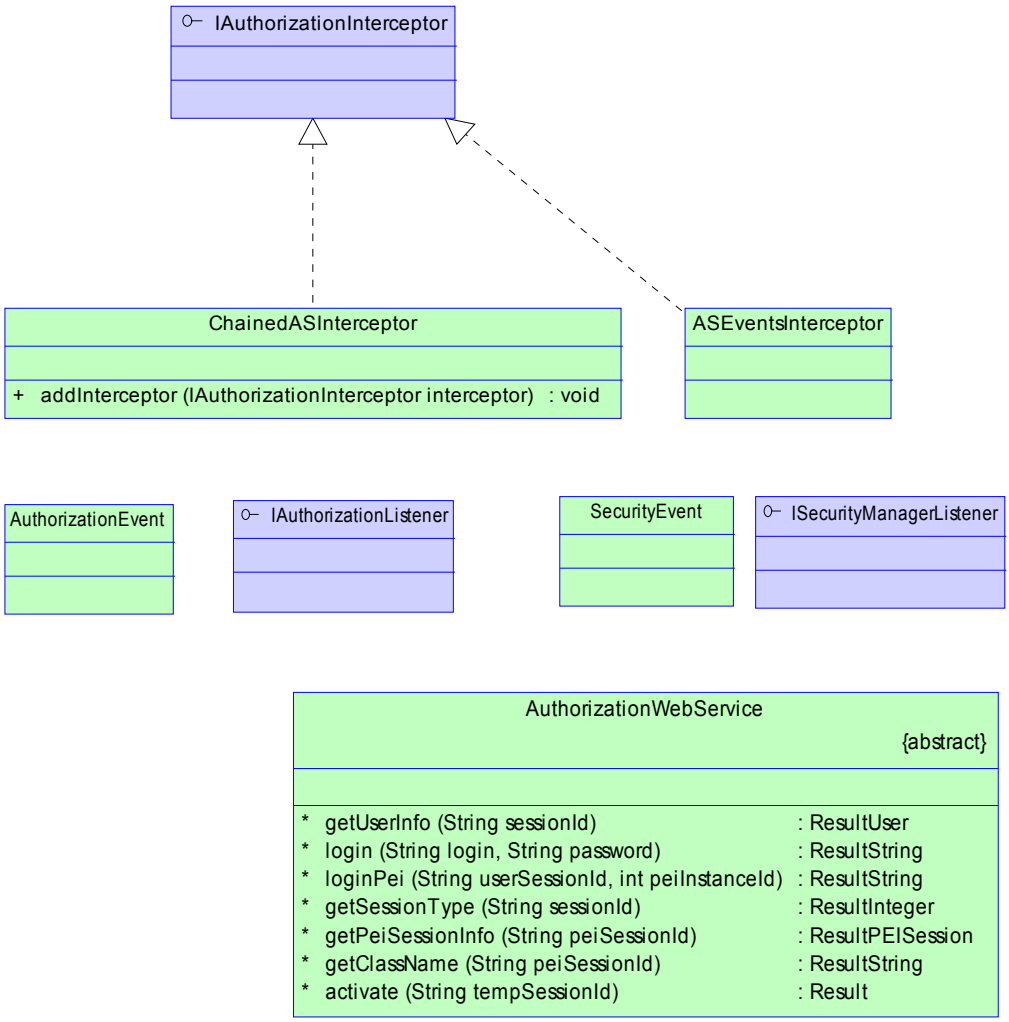


Рисунок 30. Программные интерфейсы сервисов модуля авторизации

7.1.4.2Описание сервисов модуля UDDI

7.1.4.2.1 Общее описание

Модуль UDDI предназначен для регистрации, хранения и модификации об организациях, их услугах, а так же технических спецификаций предоставляемых услуг.

В состав OracleAS-10g входит реализация реестра UDDI. В рамках работ 2004 года разработан локальный Java API сервиса UDDI. Удаленный SOAP-интерфейс соответствует спецификации UDDI v2.0. Таким образом Oracle UDDI не доступен напрямую ни другим сервисам Среды, ни сторонним организациям.

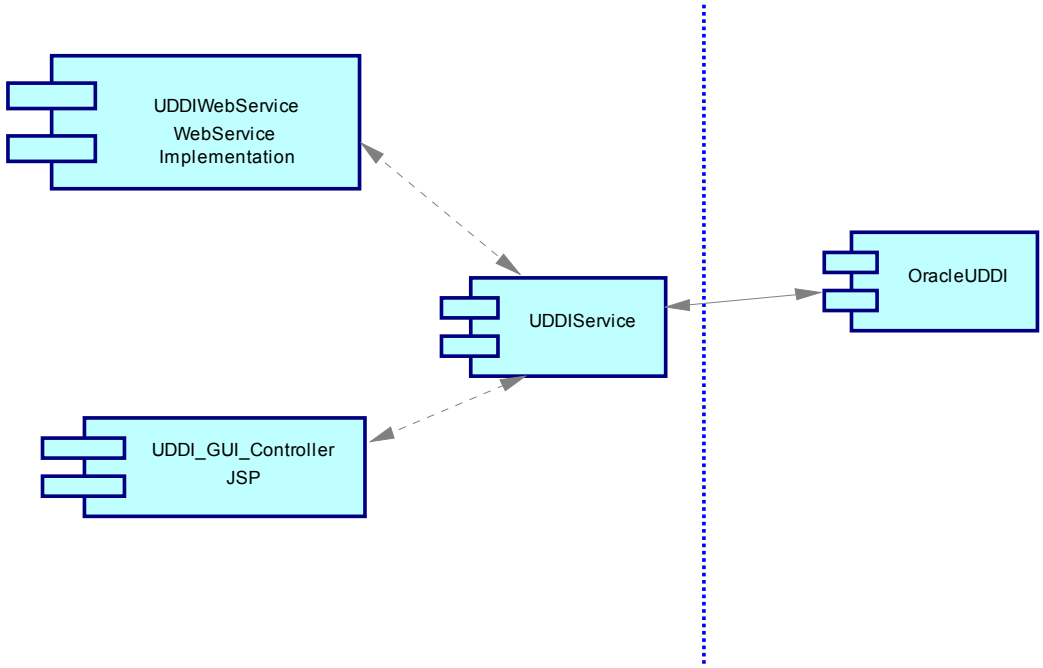


Рисунок 31. Компоненты модуля UDDI и их связи

Требования по взаимодействию с другими подсистемами ядра:

- проверять наличие типов данных, заявленных в публикуемом сервисе (путем проверки передаваемого в виде URL-адреса WSDL-описания) в репозитарии Среды, в случае отсутствия описания нужно отказать в регистрации сервиса;
- авторизацию пользователей, используя API сервиса авторизации
- добавление данных об организации в подсистему ССК при публикации новой организации;
- проверять наличие указанных при публикации классификаций в ССК и при их отсутствии отказать в публикации.

7.1.4.2.2 Программные интерфейсы

Модуль UDDI предоставляет следующие интерфейсы для других подсистем: *IUDDIService* и *UDDIInterceptor*.

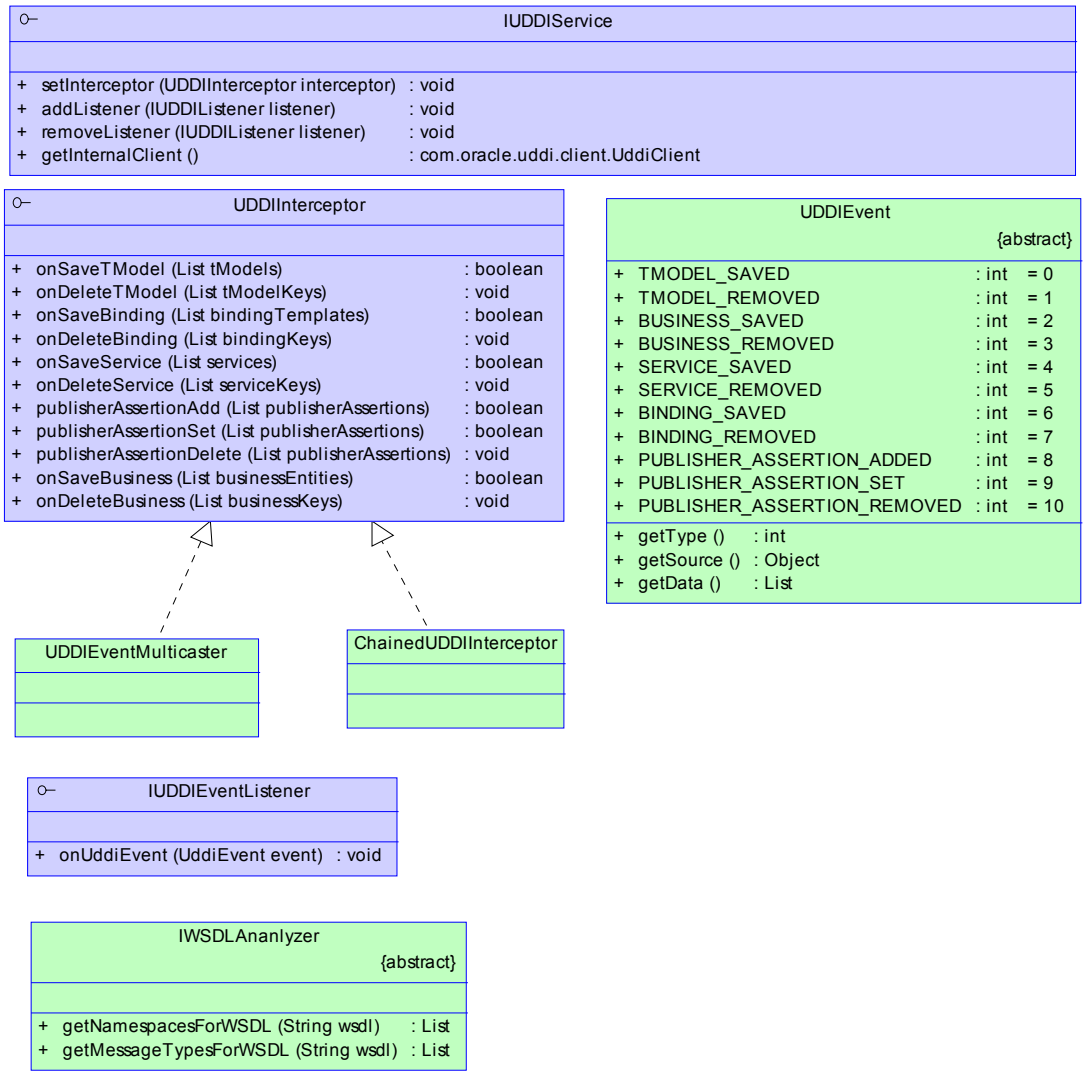


Рисунок 32. Схема основных классов и интерфейсов модуля UDDI

7.1.4.2.3 Интерфейс пользователя

Исполнительная среда сервиса, клиентской и административной части пользовательского интерфейса – сервер Tomcat 5.

Интерфейс пользователя предоставляет возможности простого и расширенного поиска организаций и услуг, интерфейс для публикации данных об организациях и услугах и выборку данных, опубликованных пользователем. При этом GUI ССК (выбор классификатора и значений классификатора) интегрировано в экраны пользовательского интерфейса. При просмотре результатов поиска предоставляется регистрационная информация об услугах и организациях, а так же, в случае, если услуга является ПЭВ, ссылка на страницу запуска ПЭВ.

7.1.4.2.4 Интерфейс администратора

Интерфейс администратора предоставляет возможность добавления, удаления и изменения опубликованных данных, доступ к настройке пользовательского интерфейса и настройкам соединения с Oracle UDDI (имя, пароль, *url* и т.п.).

7.1.4.2.5 Взаимодействие с модулем управления классификаторами

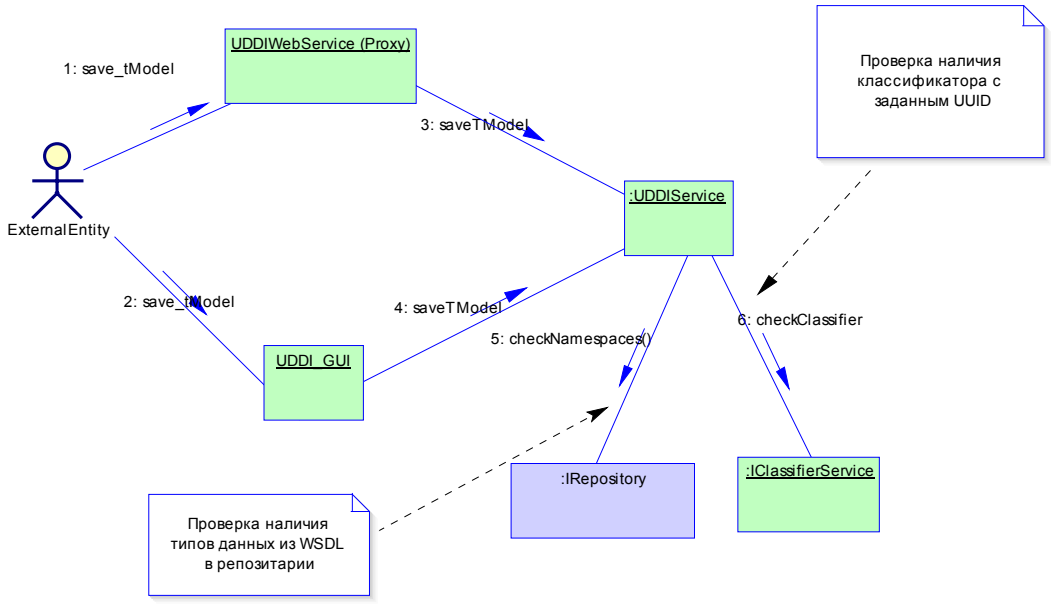


Рисунок 33. Схема взаимодействия UDDIService с подсистемой репозитария и ССК

7.1.4.3Описание сервисов модуля исполнения ПЭВ

7.1.4.3.1 Общее описание

Модуль Среды Электронного Взаимодействия (СЭВ) BPEL4WS обеспечивает размещение описаний Процедур Электронного Взаимодействия (ПЭВ), запуск, исполнение и мониторинг хода выполнения ПЭВ.

Также, в целях проверки соответствия сценариев и интерфейсов ПЭВ предъявляемым к ним требованиям модуль исполнения ПЭВ обеспечивает:

- поддержку определения контрольных точек на маршруте ПЭВ в ходе разработки XML-описаний ПЭВ;
- средства автоматической проверки корректности WSDL-описания ПЭВ в качестве web-сервиса при его регистрации в реестре UDDI Среды;
- средства автоматической проверки соответствия WSDL-описаний формальным требованиям, предъявляемым к описанию процедур электронного взаимодействия;

- средства автоматической проверки соответствия сценариев ПЭВ требованиям, связанным с контролем последовательности операций по управлению сессиями;
- средства автоматической проверки соответствия сценариев ПЭВ требованиям, определяющим последовательность команд по управлению транзакциями и ограничения на их применение;
- средства автоматической проверки соответствия сценариев ПЭВ требованиям, касающимся последовательности обращений к методам инициализации web-сервисов и завершения их работы.

Основные компоненты модуля:

- в качестве базового исполнительного механизма используется Oracle BPEL Process Manager (BPEL-PM);
- сервис ядра СЭВ – *BPELService*, обеспечивает программный интерфейс к BPEL-PM;
- сервис исполнения;
- веб-сервис *BPELProxy*, обеспечивающий передачу сообщений извне;
- интерфейс пользователя;
- интерфейс администратора.

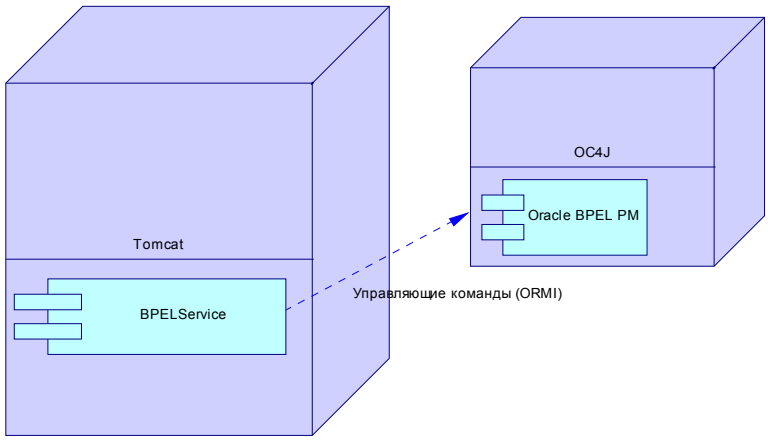


Рисунок 34. Схема размещения компонент модуля исполнения

7.1.4.3.2 Программные интерфейсы

7.1.4.3.2.1 BPELService

BPELService представляет собой обертку над BPEL-PM API. При регистрации нового BPEL4WS-процесса сервис автоматически проверяет корректность WSDL-описания и обеспечивать регистрацию ПЭВ в UDDI-реестре.

7.1.4.3.2.2 Сервис исполнения

Web-сервис BPELProxy обеспечивает SOAP-интерфейс для обращения других Web-компонент, участвующих в процессе исполнения ПЭВ.

Для разбиения единого процесса на этапы, по которым возможно оценить его макросостояние, администратором Среды при описании сценария ПЭВ вводятся горизонтальная и вертикальная метрики. Горизонтальная метрика характеризует степень завершенности процесса, а вертикальная дает представление об ответственном объекте взаимодействия на текущем этапе. Для реализации горизонтальной метрики вводится набор стадий, а для вертикальной – набор статусов. Процесс наложения метрики представляет создание статусов и стадий, и последующее распределение действий по ним. Одной стадии могут соответствовать несколько статусов, детально описывающих состояние экземпляра ПЭВ в каждый момент его выполнения. Переход ПЭВ в финальное состояние также сопровождается присвоением результату некоторого статуса, обычно «успешное завершение» или «останов по указанию пользователя».

Пример схемы макросостояний процедуры:

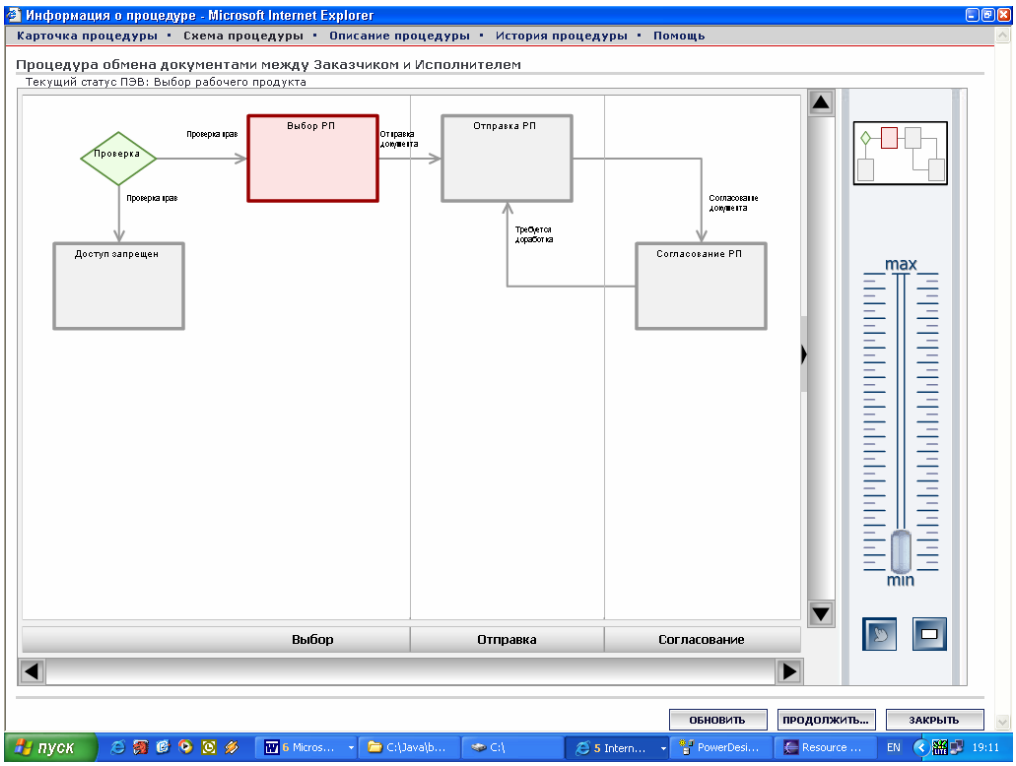


Рисунок 35. Интерфейс просмотра метрик ПЭВ

Пример xml-описания макросостояний

```
<?xml version="1.0" encoding="Windows-1251"?>
<peiClass name="ProductTrackingDigitalProcess2">
```

```
    <description>Процедура обмена документами между Заказчиком и
Исполнителем</description>
    <help>Процедура контроля завершения проектов состоит из двух основных шагов.
На первом шаге происходит получение пользователем перечня всех  реализуемых
проектов с указанием их состояний и выделением  контролируемых им проектов, а такж
выбор пользователем проекта,  мониторинг которого должен быть начат. На втором шаг
выбранный проект  устанавливается на контроль и после завершения его выполнения
пользователю отсылается соответствующее сообщение. Снятие проекта с  контроля
осуществляется путем принудительного завершения работы  экземпляра процедуры
контроля.  &lt;br>&lt;br>  В карточке процедуры электронного взаимодействия
отображается текущее состояние ее выполнения. Если состояние выполнения
подразумевает участие пользователя, то  необходимо нажать кнопку
&quot;Продолжить&quot; внизу страницы и ввести необходимую информацию.  На схеме
процедуры электронного взаимодействия красным цветом выделяется текущая стадия;
если ее выполнение требует участия пользователя, то для перехода к вводу информаци
достаточно нажать на этом прямоугольнике.</help>
    <macrostate coordX="1.0" coordY="0.0" invisible="1"
logpointBeginName="Logpoint.Init.Begin" logpointEndName="Logpoint.Init.End"
name="Инициализация" typeName="Вызов специализированного сервиса"/>
    <macrostate coordX="1.0" coordY="0.0" invisible="0"
logpointBeginName="Logpoint.Select.Begin" logpointEndName="Logpoint.Select.End"
name="Выбор РП" typeName="Вызов сервиса">
        <businessLogpoint name="LogpointBusiness.Select"/>
        <businessLogpoint name="LogpointBusiness.Select0"/>
    </macrostate>
    <macrostate coordX="2.0" coordY="1.0" invisible="1"
logpointBeginName="Logpoint.Exit.Begin" logpointEndName="Logpoint.Exit.End"
name="Отмена" typeName="Вызов специализированного сервиса"/>
    <macrostate coordX="2.0" coordY="0.0" invisible="0"
logpointBeginName="Logpoint.Send.Begin" logpointEndName="Logpoint.Send.End"
name="Отправка РП" typeName="Вызов сервиса">
        <businessLogpoint name="LogpointBusiness.Send"/>
        <businessLogpoint name="LogpointBusiness.Send0"/>
    </macrostate>
    <macrostate coordX="3.0" coordY="1.0" invisible="0"
logpointBeginName="Logpoint.Work.Begin" logpointEndName="Logpoint.Work.End"
name="Согласование РП" typeName="Вызов сервиса">
        <businessLogpoint name="LogpointBusiness.Work"/>
    </macrostate>
```

```
    <macrostate coordX="5.0" coordY="0.0" invisible="1"
logpointBeginName="Logpoint.Finish.Begin" logpointEndName="Logpoint.Finish.End"
name="Завершение работы" typeName="Вызов специализированного сервиса"/>
    <macrostate coordX="0.0" coordY="0.0" invisible="0"
logpointBeginName="Logpoint.BeforeInit.Begin"
logpointEndName="Logpoint.BeforeInit.End" name="Проверка прав" typeName="Условие"/
    <macrostate coordX="0.0" coordY="1.0" invisible="0"
logpointBeginName="Logpoint.AccessDenied.Begin"
logpointEndName="Logpoint.AccessDenied.End" name="Доступ запрещен" typeName="Вызов
сервиса"/>
    <status name="Выбор рабочего продукта"/>
    <status name="Отправка документа Заказчику"/>
    <status name="Согласование документа Заказчиком"/>
    <status name="Требуется доработка документа"/>
    <status name="Завершение работы над продуктом"/>
    <status name="Отмена процедуры"/>
    <status name="Проверка прав"/>
    <transitions leftMc="Инициализация" name="toSelect" rightMc="Выбор РП"
statusName="Выбор рабочего продукта">
        <description/>
    </transitions>
    <transitions leftMc="Выбор РП" name="toSend" rightMc="Отправка РП"
statusName="Отправка документа Заказчику">
        <description/>
    </transitions>
    <transitions leftMc="Выбор РП" name="toExit" rightMc="Отмена"
statusName="Отмена процедуры">
        <description/>
    </transitions>
    <transitions leftMc="Проверка прав" name="toInit" rightMc="Выбор РП"
statusName="Проверка прав">
        <description/>
    </transitions>
    <transitions leftMc="Отправка РП" name="toWork" rightMc="Согласование РП"
statusName="Согласование документа Заказчиком">
        <description/>
    </transitions>
    <transitions leftMc="Согласование РП" name="toSelect2" rightMc="Отправка РП"
statusName="Требуется доработка документа">
```

```
        <description/>
    </transitions>
    <transitions leftMc="Согласование РП" name="toFinish" rightMc="Завершение
работы" statusName="Завершение работы над продуктом">
        <description/>
    </transitions>
    <transitions leftMc="Проверка прав" name="toAccessDenied" rightMc="Доступ
запрещен" statusName="Проверка прав">
        <description/>
    </transitions>
    <stage coordX="0.0" name="Выбор">
        <stageSet mcName="Инициализация"/>
        <stageSet mcName="Отмена"/>
        <stageSet mcName="Выбор РП"/>
    </stage>
    <stage coordX="0.0" name="Отправка">
        <stageSet mcName="Отправка РП"/>
    </stage>
    <stage coordX="0.0" name="Согласование">
        <stageSet mcName="Завершение работы"/>
        <stageSet mcName="Согласование РП"/>
    </stage>
</peiClass>
```

В BPEL4WS-описатель вставляются элементы BPEL4WS invoke имеющие имена, соответствующие шаблонам: *Logpoint.<name>*, а так же *LogpointBusiness.<name>*.

Например:

```
<invoke    name="Logpoint.BeforeInit.Begin"
partnerLink="execService"
portType="es:PEIExecutionService"
operation="log"
inputVariable="logReq"
outputVariable="logResp"/>
```

Существующий парсер BPEL4WS-описателя составляет список Logpoint'ов и предоставляет их в окошке выбора логпойнта в редакторе метрик, поэтому важно соблюдать соглашения по именам BPEL4WS-блоков

Более полные требования к оформлению системных вызовов в BPEL4WS приведены в разделе «6.7.1.5 Описание системных переменных, присутствующие в описании ПЭВ, и правил оформления вызовов системных WEB-сервисов».

Операция `log` применяется для сообщения сервису исполнения отметки о прохождении точки (logpoint) на основании чего отмечаются переходы из одного макросостояния в другое.

Операция `businessLogRequest` используется для передачи строки с пояснением, описывающем произошедший переход.

Операция `waitForUserRequest` используется для сообщения сервису исполнения URL к форме ввода данных.

Java API существующего сервиса исполнения (см. также WSDL-описание) имеет вид:

```
public interface PEIExecutionService extends Remote {

    /** метод осуществляет логирование. peiSessionId - Id сессии, logpointName - имя логпоинта*/
    public Result log(String peiSessionId, String logpointName) throws RemoteException;

    /** создает экземпляр ПЭВа. userSessionId - Id сессии, classId - класс ПЭВа.*/
    public ResultID createInstance(String userSessionId, int classId) throws RemoteException;

    /** Завершает сессию ПЭВа. peiSessionId - Id сессии */
    public Result logout(String peiSessionId) throws RemoteException;

    /** Метод переводит ПЭВ в состояние ожидания ввода пользователя.
     *  peiSessionId - Id сессии, url - URL по которому будут вводиться данные */
    public Result waitForUser(String peiSessionId, String url) throws RemoteException;

    /** Проверяет состояние ПЭВа. Возвращает
     * 0=работа
     * 1=ожидание ввода
     * 2=останов
```

```
     * 3=завершено
    */
    public ResultInteger checkSystemState(String peiSessionId) throws
RemoteException;

    /** Устанавливает флаг останова. ПЭВ остановится как только сможет
     * peiSessionId - Id сессии */
    public Result setStopable(String peiSessionId) throws RemoteException;

    /** Сбрасывает флаг останова. peiSessionId - Id сессии */
    public Result unsetStopable(String peiSessionId) throws RemoteException;

     /** Возвращает информацию об экземпляре ПЭВа. peiSessionId - Id сессии */
    public ResultInstanceInfo getInstance(String peiSessionId) throws
RemoteException;

    /** Возвращает информацию обо всех запущенных ПЭВах в рамках данной сессии.
peiSessionId - Id сессии*/
    public ResultInstanceInfo getInstances(String userSessionId) throws
RemoteException;

    /** Возвращает информацию обо всех запущенных ПЭВах. peiSessionId - Id сессии*
    public ResultInstanceInfo getAllInstances(String userSessionId) throws
RemoteException;

    /** Возвращает информацию обо всех запущенных пользователем ПЭВах */
    public ResultInstanceInfo getAllInstancesForUser(String userSessionId, int
userId) throws RemoteException;

    /** Возвращает детальную информацию об экземпляре ПЭВ. Метод кэширует
результаты
     *  @see ru.ifirst.erussia.es.webservice.util.Graph
    */
    public ResultGraph getDetailedInstanceInfo(String peiSessionId) throws
RemoteException;

    /** Делает тоже самое, что и метод getDetailedInstanceInfo только без
перекодировки спец. символов */
```

```
    public ResultGraph getDetailedInstanceInfoClear(String peiSessionId) throws
RemoteException;

    /** Возвращает статусы, стадии и их координаты для данного экземпляра ПЭВа */
    public ResultMetrics getInstanceMetrics(String peiSessionId) throws
RemoteException;

    /** Возвращает историю изменения стадий ПЭВа */
    public ResultStageHistory getInstanceStageHistory(String peiSessionId) throws
RemoteException;

    /** Возвращает историю изменения статусов ПЭВа */
    public ResultStatusHistory getInstanceStatusHistory(String peiSessionId) throw
RemoteException;

    /** Посылает сигнал останова экземпляру ПЭВа */
    public Result sendStopSignal(String peiSessionId) throws RemoteException;

    /** Продолжает работу ПЭВа после останова */
    public Result continueWork(String peiSessionId) throws RemoteException;

    /** Запускает ПЭВ на выполнение */
    public ResultID startPEI(String userSessionId, int classId) throws
RemoteException;

    public Result businessLog(String peiSessionId, String logpointName, String
value) throws RemoteException;

    public ResultUnitedHistory getHistories(String peiSessionId) throws
RemoteException;

    public ResultUnitedHistory getCurrentInfo(String peiSessionId) throws
RemoteException;

    public Result writeUnitedHistory(String peiSessionId) throws RemoteException;

    public ResultString getHelp(String peiSessionId) throws RemoteException;
}
```

7.1.4.3.3 Интерфейс пользователя

Интерфейс пользователя модуля исполнения ПЭВ к сервису исполнения ПЭВ обеспечивает пользователям субъектов взаимодействия как возможность внесения необходимых исходных данных в параметры ПЭВ во время его запуска, так и для ввода и (или) коррекции этих данных в процессе исполнения ПЭВ. При этом GUI модуля исполнения обеспечивает возможность использования информации из подсистемы классификаторов в процессе заполнения значений параметров ПЭВ.

Кроме того, GUI модуля исполнения ПЭВ предоставляет субъекту взаимодействия информацию о состоянии активизированных им ПЭВ и возможность частичного или полного останова выполнения запущенного ПЭВ.

Авторизация пользователя при работе с GUI модуля исполнения ПЭВ осуществляется при входе пользователя в портал.

7.1.4.3.4 Интерфейс администратора

Интерфейс администратора обеспечивает следующие возможности:

- загрузки описаний процессов;
- редактирования метрик для сервиса исполнения;
- запуск экземпляра ПЭВ через заполнение параметров HTML-формы или через ввод инициирующего XML-сообщения;
- просмотра контекста выполнения экземпляра ПЭВ;
- просмотр значений переменных контекста выполнения экземпляра ПЭВ;
- приостановка и продолжение выполнения экземпляра ПЭВ;
- отмена и удаление экземпляра ПЭВ;
- запуск нагрузочных тестов по конкретному ПЭВ и просмотр статистики.

7.1.4.3.5 Диаграммы взаимодействия

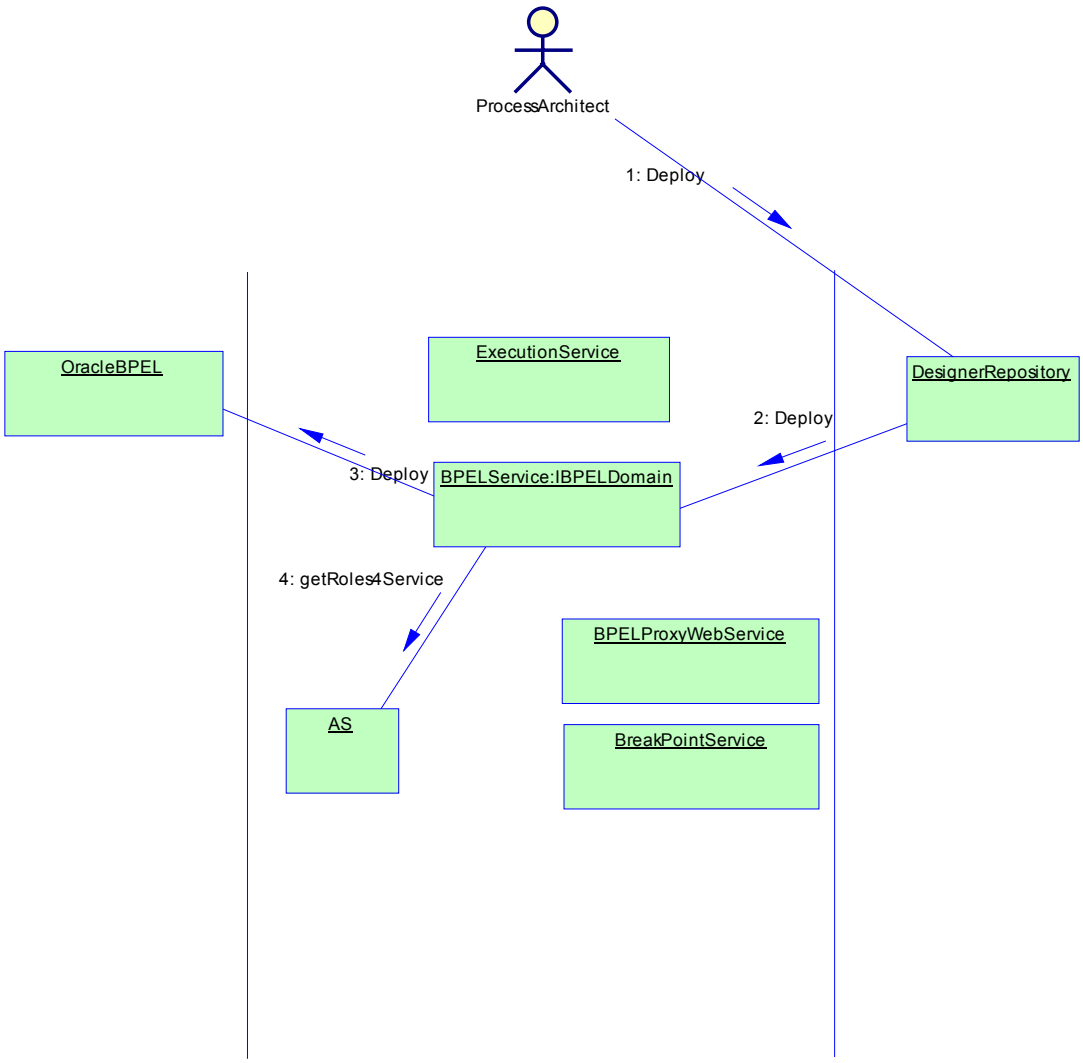


Рисунок 36. Схема взаимодействий при размещении процесса

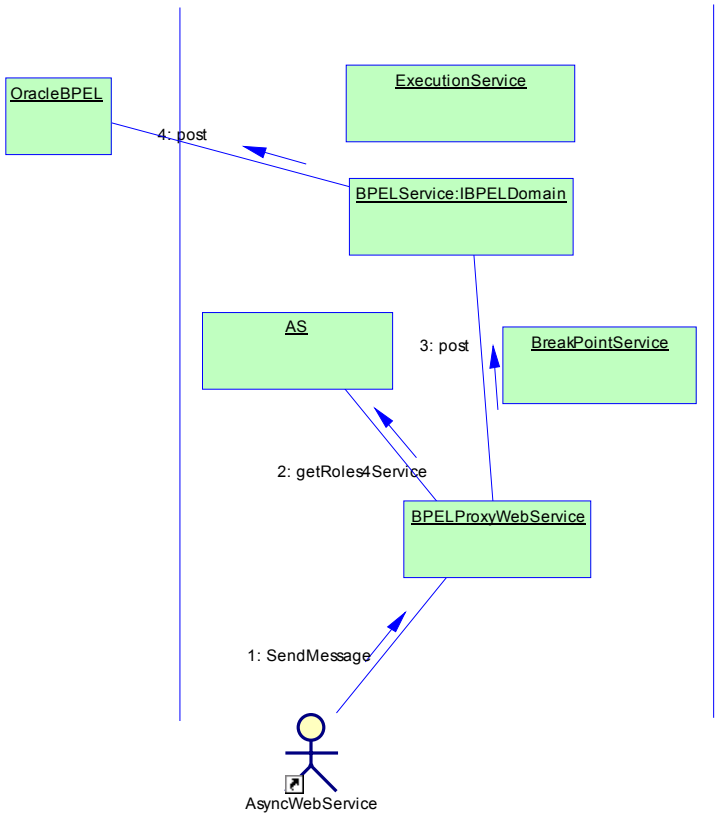


Рисунок 37. Схема взаимодействия с асинхронным сервисом

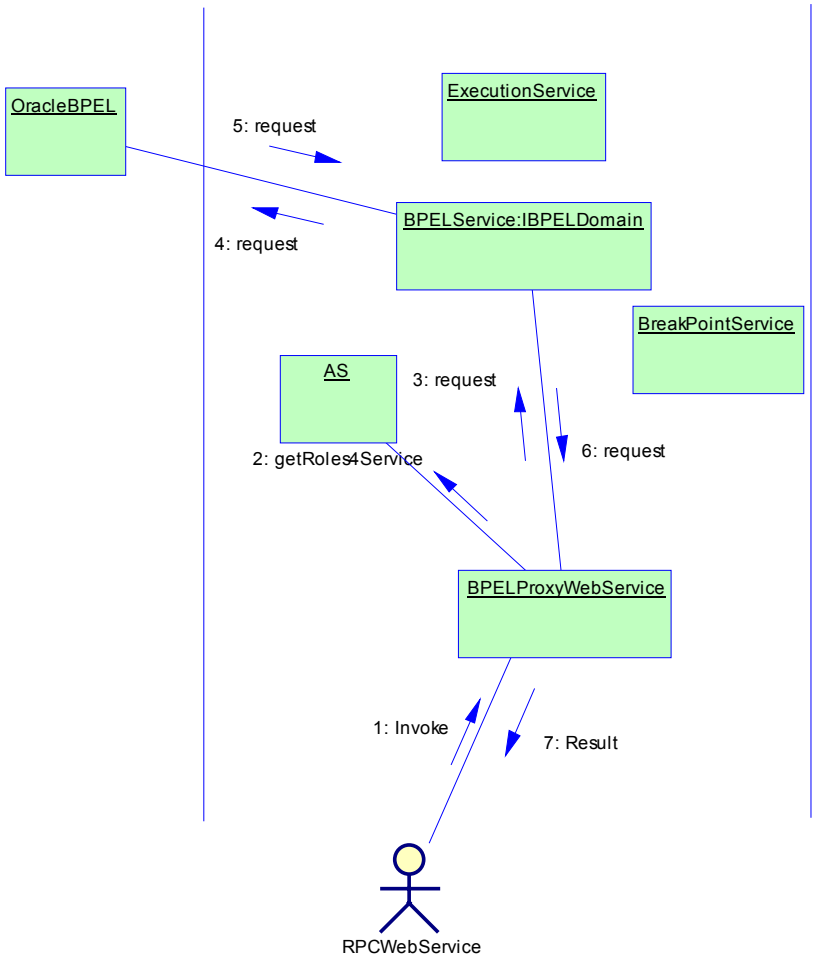


Рисунок 38. Схема взаимодействия с синхронным сервисом

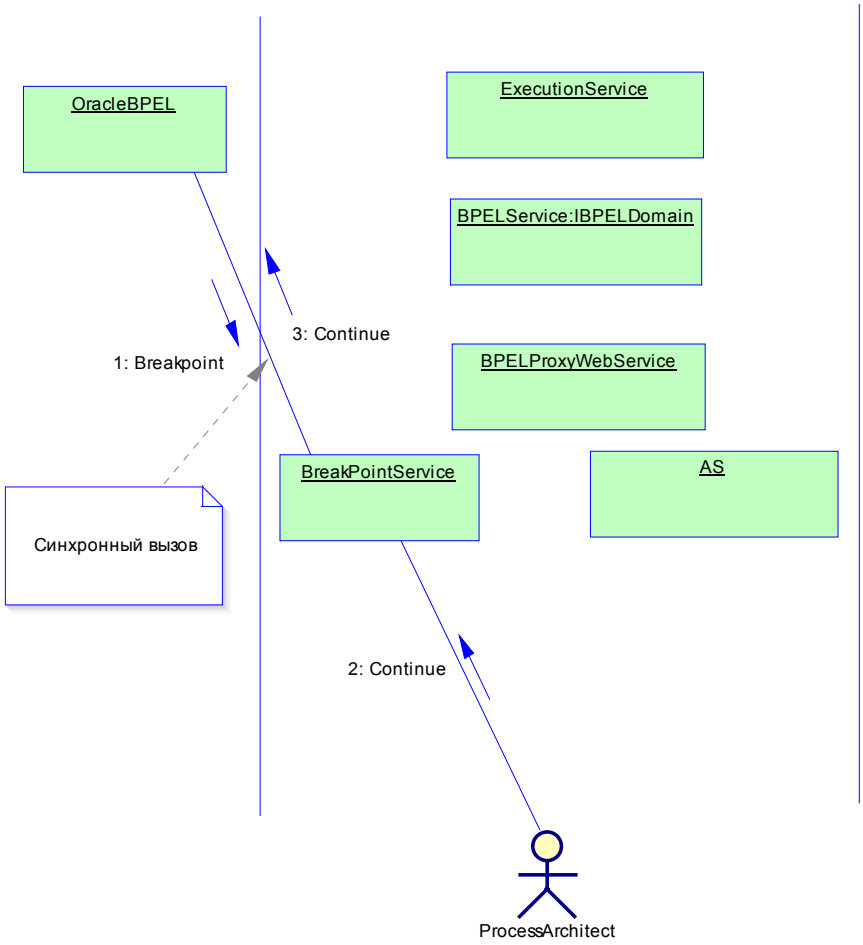


Рисунок 39. Схема реализации точек остановки

7.1.4.4Описание сервисов модуля классификаторов

7.1.4.4.1 Общее описание сервиса

Модуль ССК является компонентом ядра Среды и предназначен для хранения и контроля целостности справочников, словарей и классификаторов, используемых при организации взаимодействия в рамках Среды, например:

- Общероссийские ССК:
  - Общероссийский классификатор отраслей народного хозяйства (ОКОНХ);
  - Общероссийский классификатор видов экономической деятельности, продукции и услуг (ОКДП);
  - Классификатор форм собственности;
  - Государственный рубрикатор научно-технической информации (ГРНТИ);
  - Товарная номенклатура внешнеэкономической деятельности (ТН ВЭД);
- Международные ССК:

- Североамериканская индустриальная классификационная система (NAICS);
- Универсальные стандартные коды по продуктам и услугам (UNSPSC);
- Географический рубрикатор ISO 3166;
- Стандартный промышленный классификатор (SIC);
- GeoWeb географический классификатор.

Сервис ССК обеспечивает:

- возможность использования универсальной структуры БД или конкретно указываемых таблиц для хранения справочников или гибридной структуры (например, часть справочников настраивается на существующие таблицы, а часть может быть динамической) - это определяется в конфигурационном файле (все справочники статической части автоматически системные);
- наличие у любого справочника полей NAME (уникальное название справочника для использования через API, совпадающее с названием таблицы для случая статических справочников), DESC (название справочника для использования в UI), ELEM_ID (поле-уникальный идентификатор элемента справочника), ELEM_NAME (поле-название элемента справочника для использования через API), ELEM_DESC (поле-название элемента справочника для использования в UI), PAR_ID (поле-идентификатор родительского элемента справочника) – может быть NULL, если справочник неиерархический, SYSTEM (флаг, означающий невозможность изменения структуры справочника);
- наличие следующих программных методов:
  - создание нового справочника;
  - переименование выбранного справочника (только если это не справочник не системный и при условии отсутствия элементов);
  - удаление выбранного справочника (только если это не справочник не системный и при условии отсутствия элементов);
  - установка системности/несистемности выбранного справочника (только если это не справочник статической части базы данных модуля);
  - выбор справочника по NAME;
  - получение NAME выбранного справочника;
  - получение DESC выбранного справочника;
  - получение наименования атрибута выбранного справочника;
  - получение типа атрибута выбранного справочника;

- переименование атрибута выбранного справочника (только если это не системный и при условии отсутствия элементов);
- изменение типа выбранного атрибута справочника (только если это не справочник не системный и при условии отсутствия элементов);
- добавление атрибута в справочник (только если это не справочник не системный и при условии отсутствия элементов);
- удаление атрибута из справочника (только если это не справочник не системный и при условии отсутствия элементов);
- добавление нового элемента в выбранный справочник;
- переименование выбранного элемента выбранного справочника;
- сохранение изменений полей выбранного элемента выбранного справочника;
- удаление элемента выбранного справочника;
- получение элемента выбранного справочника по значению ID;
- получение коллекции из элементов выбранного справочника по значению PAR_ID;
- получение коллекции из всех элементов выбранного справочника
- получение коллекции из элементов выбранного справочника по значению NAME;
- получение коллекции из элементов выбранного справочника по значению указанного атрибута;
- очистка всех элементов выбранного справочника;
- получение ID элемента выбранного справочника;
- получение NAME элемента выбранного справочника;
- получение PAR_ID элемента выбранного справочника;
- получение атрибута элемента выбранного справочника;
- установка NAME элемента выбранного справочника;
- установка атрибута элемента выбранного справочника;
- установка родительского элемента для элемента выбранного справочника;

- наличия механизма закладок, по которым распределяются атрибуты справочников (атрибут справочника может находиться в нескольких или всех закладках);
- возможность организации поиска по значению атрибутов справочников со следующими спецификаторами: равно, не равно, больше, меньше, включает, содержится, начинается с, кончается на, меньше или равно, больше или равно;
- наличие следующих программных методов:

- создание новой закладки для выбранного справочника;
- получение списка всех закладок для выбранного справочника;
- получение порядкового номера закладки для выбранного справочника;
- установка порядкового номера закладки для выбранного справочника;
- переименование закладки для выбранного справочника;
- удаление закладки для выбранного справочника;
- получение списка атрибутов, входящих в закладку выбранного справочника;
- установка списка атрибутов, входящих в закладку выбранного справочника;
- помещение атрибута выбранного справочника в закладки (одну, несколько, все);
- поиск элементов выбранного справочника по значениям атрибутов с учетом спецификаторов поиска;

- наличие периодических реквизитов справочника (по которым хранится история изменений реквизита);
- наличие системных/несистемных реквизитов справочника (для которых невозможно менять название атрибута и его тип);
- наличие связей между справочниками (элемент-владелец тот же элемент-родитель, но из другого справочника);
- наличие следующих программных методов:
  - получение периодического атрибута элемента выбранного справочника;
  - установка периодического атрибута элемента выбранного справочника;
  - получение истории изменений атрибута элемента выбранного справочника;
  - установка системности/несистемности атрибута выбранного справочника;
  - получение коллекции элементов выбранного справочника по справочнику–владельцу и значению NAME элемента справочника-владельца;
  - получение коллекции элементов выбранного справочника по справочнику–владельцу значению ID элемента справочника-владельца;
  - получение коллекции элементов выбранного справочника по справочнику–владельцу значению атрибута элемента справочника-владельца.

7.1.4.4.2 Программные интерфейсы

Наличие основных интерфейсов:

1. Classifier – интерфейс справочника
   - String getName() – получение имени справочника;

- void setName(String name) – переименование справочника (только если справочник несистемный);
- String getUUID() – получение UUID классификатора, с которым он зарегистрирован в UDDI-реестре (необязательный атрибут);
- void setUUID(String uuid) – присвоение UUID классификатору;
- boolean isSystem() – проверка системности;
- void setSystem(boolean system) – установка системности;
- boolean isEmpty() – проверка на наличие значений;
- String getDescription() – получение описания справочника;
- void setDescription(String description) – установка описания справочника (для несистемного);
- void addValue(ClassifierValue value) – добавить значение;
- void deleteValue(ClassifierValue value) – удалить значение;
- ClassifierValue getValueById(long valueId) – получить значение по id;
- List getValues() – получить все значения;
- List getValuesByParentId(long valueParentId) – получить значения по id родительской записи;
- List getValuesByName(String name) – получить все значения по данному названию;
- List getValuesByAttributeValue(ClassifierAttribute attribute, String value) – получить все значения справочника по значению указанного атрибута;
- void addAttribute(ClassifierAttribute attribute) – добавить атрибут к справочнику (если справочник не системный и не пустой);
- public ClassifierAttribute getAttributeByName(String name) – получить атрибут справочника по названию;
- void deleteAttribute(ClassifierAttribute attribute) – удалить атрибут из справочника (если справочник не системный и не пустой);
- List getAttributes() – получить список атрибутов справочника;
- void clean() – очистить справочник;
- int size() – получить количество элементов справочника.
- List getBookmarks() – список закладок;
- ClassifierBookmark getBookmarkByName(String bookmarkName) – получить закладку на названию;

- void addBookmark(ClassifierBookmark classifierBookmark) – добавить закладку к справочнику;
- void removeBookmark(ClassifierBookmark classifierBookmark) – удалить закладку из справочника.

2. ClassifierValue – интерфейс элемента (значения) справочника
   - Classifier getClassifier() – получить справочник данного элемента;
   - long getId() – идентификатор;
   - String getName() – название;
   - void setName(String name) – установить новое название;
   - String getDescription() – описание;
   - void setDescription(String description) – установить новое описание;
   - ClassifierValue getParent() – получить родительское значение;
   - List getChildrens() – получить список дочерних значений;
   - List getAttributeValues() – получить значения атрибутов;
   - ClassifierAttributeValue getAttributeValue(ClassifierAttribute attribute) – получить значение атрибута;
   - void setAttributeValue(ClassifierAttributeValue attributeValue) – установить значение атрибута.
3. ClassifierAttribute – интерфейс атрибута справочника
   - String getName() – название;
   - void setName(String name) – установить название;
   - ClassifierAttributeType getType() – тип атрибута;
   - String getTypeName() – название типа атрибута;
   - void setType(String typeName) – установить тип атрибута по названию;
   - void changeType(String typeName) – сменить тип атрибута;
   - Classifier getClassifier() – получить справочник для атрибута;
   - void upPriority() – повысить «приоритет» атрибута. Влияет на то, в каком порядке можно получить атрибуты из справочника.
   - ClassifierBookmark getBookmark() – получить закладку для атрибута
4. ClassifierAttributeType – интерфейс типа атрибута справочника
   - String getName() – название типа;
   - boolean isValide(String value) – проверить значение на валидность данному типу атрибута.
5. ClassifierAttributeValue – интерфейс значения атрибута справочника

- ClassifierAttribute getAttribute() – атрибут;
- String getValue() – значение;
- void setValue(String value) – установить значение;
- ClassifierValue getClassifierValue() – значение справочника для значения атрибута.

IClassifierService – сервис-контроллер классфикаторов

- List getClassifiers() – получить список справочников;
- void createClassifier(Classifier classifier, boolean registerInUDDI) – создать справочник, если второй параметр имеет значение *true* – зарегистрировать классификатор в UDDI реестре;
- deleteClassifier(Classifier classifier) – удалить справочник;
- Classifier getClassifierByName(String name) – получить справочник по названию;
- Classifier getClassifierByUUID(String uuid) – получить справочник по его UUID (пустые строки не допускаются);
- Classifier constructClassifier(String name, String description, boolean system) – сконструировать справочник для последующего создания;
- ClassifierValue constructValue(Classifier classifier, ClassifierValue parentValue, String name, String description) – сконструировать значение справочника для последующего создания;
- ClassifierAttributeType getAttibuteType(String typeName) – получить тип атрибута по названию;
- ClassifierAttribute constructAttribute(Classifier classifier, ClassifierAttributeType attributeType, String name) – сконструировать атрибут;
- ClassifierAttributeValue constructAttributeValue(ClassifierValue classifierValue, ClassifierAttribute attribute, String value) – сконструировать значение атрибута
- ClassifierBookmark constructBookmark(Classifier classifier, String name) – сконструировать закладку.

ClassifierBookmark – интерфейс закладки

- String getName() – название;
- void setName(String bookmarkname) – установить название;
- List getClassifierAttributes() – получить список атрибутов для данной закладки;
- Classifier getClassifier() – получить справочник;

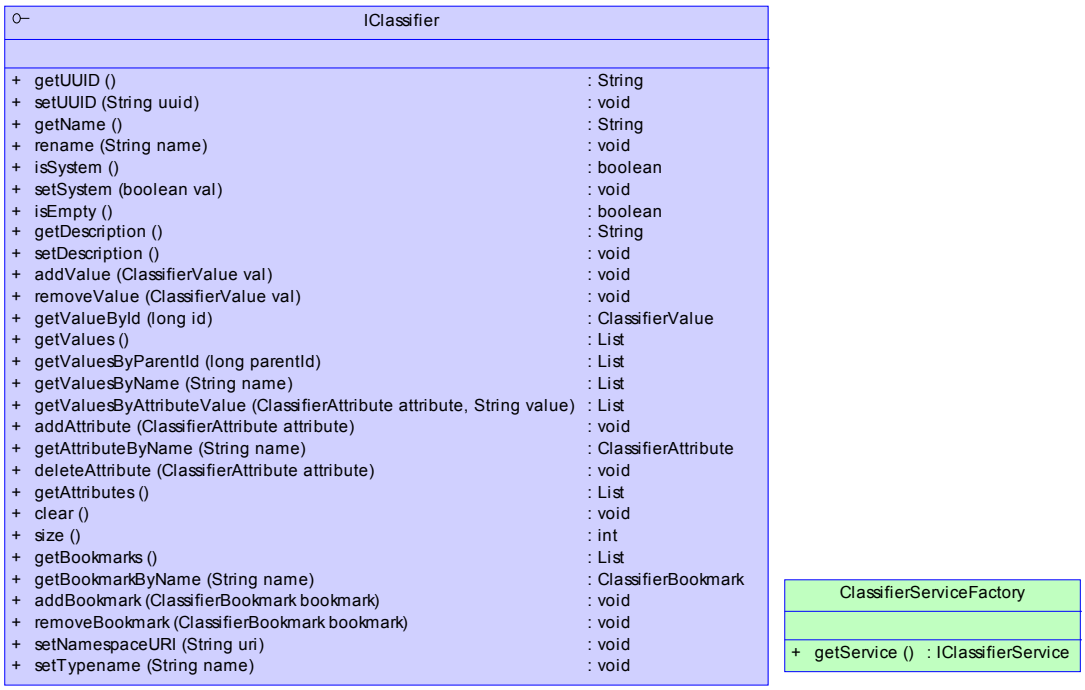
- void addAttribute(ClassifierAttribute classifierAttribute) – добавить атрибут к закладке;
- void removeAttribute(ClassifierAttribute classifierAttribute) – удалить атрибут из закладки;
- void setGeneral() – сделать закладку закладкой по-умолчанию;
- boolean isGeneral() – проверить на то, является ли закладка закладкой по-умолчанию;
- boolean getGeneral() – аналогично isGeneral;
- void upPriority() – повысить «приоритет» закладки. Влияет на то, в каком порядке можно получить закладки из справочника;

Модификация (добавление методов) в следующие интерфейсы:

Searcher

- static Set getClassifierValues(Classifier classifier, String queryString) – получить все значения справочника, удовлетворяющие запросу.

Описание статических справочников производится в настроечном файле classifier-config.xml.



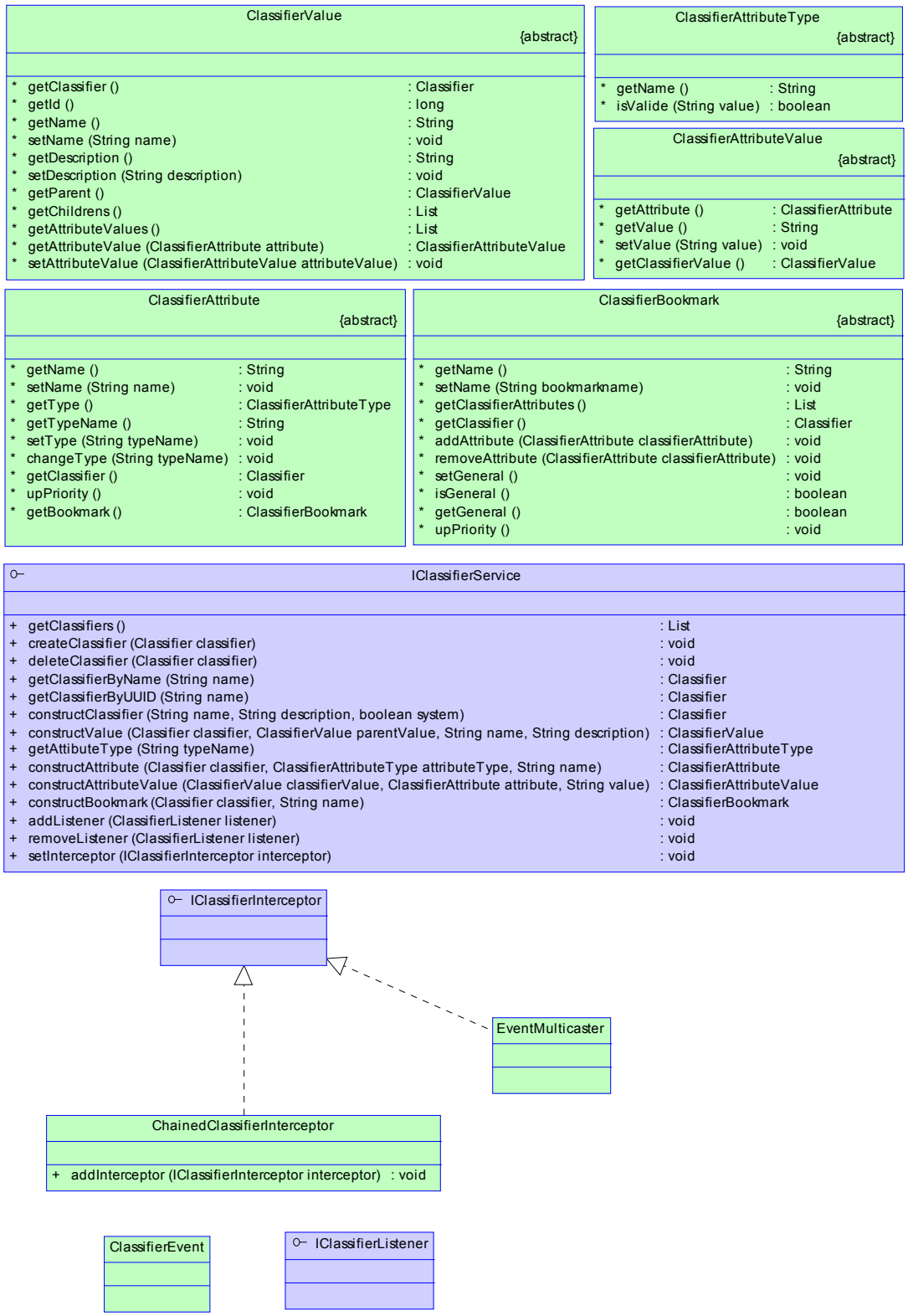


Рисунок 40. Диаграмма классов подсистемы ССК

7.1.4.4.3 Интерфейс администратора

Интерфейс администратора подсистемы ССК предназначен для управления используемыми в Среде словарями, справочниками и классификаторами, хранящимися в подсистеме ССК. Управление осуществляется посредством API подсистемы ССК.

ИА подсистемы ССК обеспечивает возможность добавления, модификации и удаления словарей, справочников и классификаторов, используемых при организации взаимодействия, как в рамках Среды, так и за ее пределами, а также получения статистических данных о составе информации, содержащейся в подсистеме ССК.

Авторизация пользователя при администрировании подсистемы ССК осуществляется модулем авторизации подсистемы безопасности Среды посредством его пользовательского интерфейса.

7.1.4.4.4 Взаимодействие с UDDI

Взаимодействие с UDDI (если требуется регистрация):

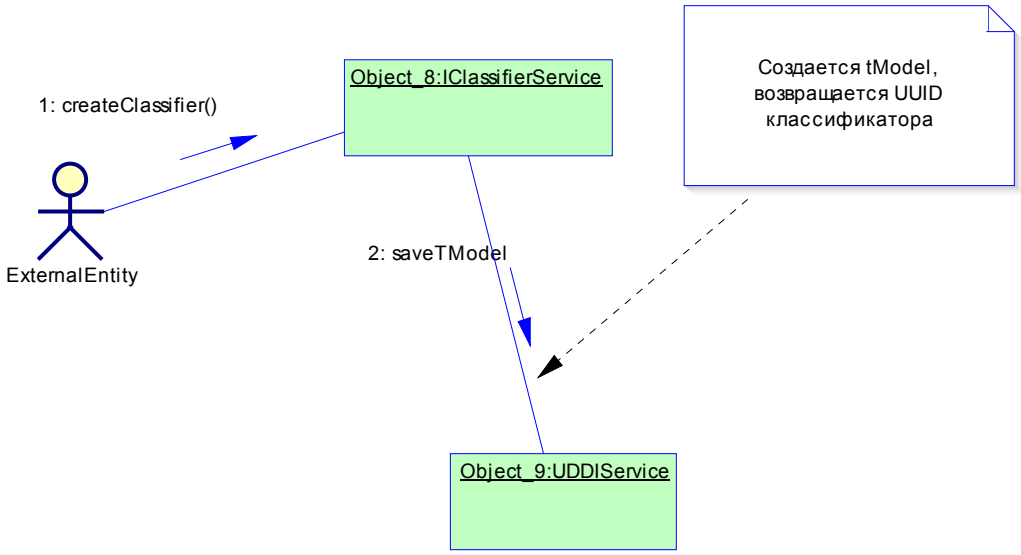


Рисунок 41. Взаимодействие ССК с UDDI

7.1.4.5Описание модуля репозитарий

7.1.4.5.1 Общее описание

Модуль «Репозитарий» предназначен для хранения информации о составе информационных ресурсов и используемых метаданных. В репозитарии хранится информация о структурах данных, используемых при реализации ПЭВ (Процедура Электронного Взаимодействия) и вызовах внутренних web-сервисов.

Информация о типе данных включает:

- описание структуры данного типа (***XSD***);
- указание типа данных, являющегося каноническим для данного типа;
- набор инструкций для преобразования данных описываемого типа к каноническому виду и обратно (***XSLT*** или процедура на языке ***Java***);

- наборы инструкций для полного или частичного преобразования данных описываемого типа к любому другому типу и, возможно, обратного преобразования. Указание имени *Java*-класса и набора параметров, осуществляющего преобразование (по умолчанию указывается имя класса `ru.ifirst.er.repository.XSLTransformer` с указанием имени ***XSLT***-файла, содержащего инструкции для преобразования);
- техническое описание типа данных и его свойств на русском языке для администратора Среды (неструктурированная справочная информация – текст, графика).

7.1.4.5.2 Программные интерфейсы

API репозитария определяет следующий набор методов:

- Boolean checkNamespace(String[] namespaces)
  Проверка наличия `namespaces` в репозитарии
- List<TypeDef> findTypes(SearchCriteria[] criteria)
  Поиск типов в репозитарии с помощью множества критериев. Возвращает набор id типов, которые удовлетворяют условиям. Условия могут накладывать ограничения на параметры типов, принадлежность к веткам ССК, принадлежность к namespaces. Условия могут быть построены на основе разнообразных операций (больше, меньше, равно, вхождение подстроки, соответствие шаблону регулярного выражения и т.д.)
- List<String> translate(String[] values, TypeDef[] types)
  Осуществляет перевод конкретных реализаций типов (значений) в значения других типов, которые указаны в массиве `types`. В случае если соответствующий тип массива `types` равен `null`, осуществляется перевод в канонический тип.

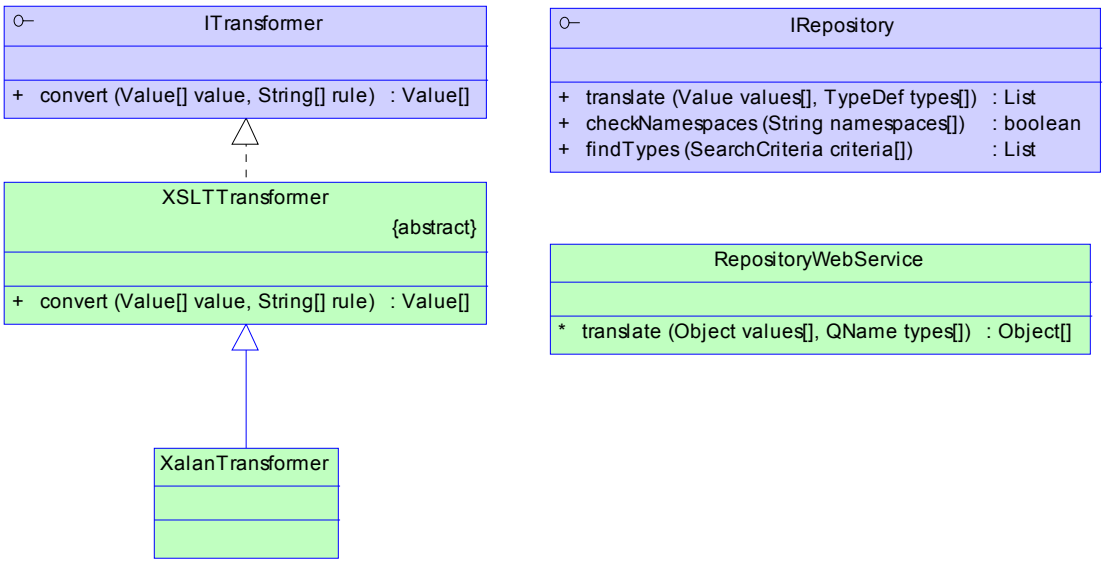


Рисунок 42. Программные интерфейсы репозитария схем данных

Модуль «Репозитарий» состоит из следующих структурных элементов, каждый из которых, в свою очередь, обладает программным интерфейсом или подробным описанием для интеграции с другими подсистемами ядра:

- база данных Oracle 9i – для хранения и обработки описаний;
- java-классы репозитария (Java API репозитария) – для обращения к репозитарию других подсистем ядра;
- web-сервис репозитария – для обращения других сервисов Среды к функциям репозитария;
- web-интерфейс репозитария, интегрируемый в интерфейсы административный портал Среды.

7.1.4.5.3 Интерфейс администратора

Web-интерфейс администратора репозитария реализует основные функции по управлению настройками и данными репозитария, а также внесению в него информации о новых типах, ее изменения и удаления. Кроме того, web-интерфейс администратора предоставить инструмент для внесения, изменения и удаления XSD-схем и обеспечивать проверку правильности их описания.

7.1.4.5.4 Взаимодействие репозитария с подсистемой ССК

В репозитарии могут быть размещены описания типов данных, являющимися значениями классификаторов (***simpleType***), поэтому при внесении такого типа в базу

репозитария, он выполнит проверку наличия такого классификатора в базе сервиса ССК и, если такой классификатор имеется, в дальнейшем проверяются (перехватываются) события подсистемы ССК с тем, чтобы блокировать попытки удаления классификаторов, объявленных в репозитарии.

7.1.5Требования к внутренним web-сервисам СЭВ

Все регистрируемые в реестре сервисов Среды электронного взаимодействия web-сервисы информационных систем субъектов взаимодействия, предназначенные для интеграции информационных систем субъектов в рамках Среды, должны удовлетворять определенному набору требований, автоматически контролируемых в процессе регистрации в реестре таких сервисов. В дальнейшем, в целях отделения их от web-сервисов, реализующих процедуры электронного взаимодействия (ПЭВ), и прочих web-сервисов, описания которых хранятся в реестре, будем называть сервисы, предоставляемые субъектами взаимодействия в целях интеграции со Средой внутренними web-сервисами Среды. Внутренние сервисы Среды в большинстве случаев реализуются в виде электронных административных регламентов (ЭАР), поэтому будем в дальнейшем использовать термин ЭАР для обозначения таких сервисов.

Используемые при описании интерфейса ЭАР (внутреннего web-сервиса Среды) типы данных должны быть получены в результате импортирования опубликованных администратором Среды пространств имен. Прямое определение типов данных в WSDL-описании web-сервисов запрещается.

Кроме того, недопустим параллельный вызов одним экземпляром ПЭВ нескольких функций одного сервиса, так как это приведет к сбоям в синхронизации сервисов. При попытке обращения к функции сервиса при незавершенном выполнении другой функции (к функциям сервиса не относятся проверка состояния сервиса, его останов, запрос дополнительных данных) результат обращения (см. ниже) должен быть «внутренняя ошибка сервиса» (RESULT_FAILED).

7.1.5.1Основные требования к web-сервисам

Разрабатываемый субъектом web-сервис должен обладать точками входа, позволяющими обратиться к его функциям. В зависимости от архитектуры web-сервиса, определяющей способ вызова его функций, описание этих точек входа осуществляется двумя способами.

1. В случае, когда обращение происходит синхронно (в частности, с использованием механизма обмена сообщениями – возврат из метода web-сервиса осуществляется после полного завершения выполнения заданной функции) используется

единственный метод, возврат из которого означает завершение работы сервиса. Первым параметром входного сообщения любого из таких методов должен быть идентификатор экземпляра ПЭВ, обращающегося к методу web-сервиса. Первым параметром выходного сообщения такого метода должен быть статус завершения работы метода (успешное завершение (RESULT_OK), внутренняя ошибка сервиса (RESULT_FAILED), экземпляр ПЭВ не имеет прав на обращение к сервису (RESULT_ACCESS_DENIED)).

```
<message name="ServiceRequest">
  <part name="eip_id" type="string"/>
   ...
</message>

<message name="ServiceResponse">
  <part name="result" type="string"/>
   ...
</message>

<operation name="Service">
  <input message="ServiceRequest"/>
  <output message="ServiceResponse"/>
</operation>
```

2. В случае, когда обращение происходит асинхронно (в частности, с использованием механизма RPC – возврат из вызванного метода web-сервиса осуществляется немедленно) используется пара методов: метод, инициирующий выполнение сервиса и метод, позволяющий получить значения выходных параметров после завершения сервиса. Кроме того, для выявления момента завершения сервиса используется периодический запрос состояния сервиса посредством метода GetStatus (см. описание ниже). Выполнение сервиса считается завершенным, когда GetStatus возвращает значение «выполнение сервиса данным экземпляром ПЭВ завершено» (STATUS_DONE); непосредственно после этого будет вызываться метод для получения значений выходных параметров.

Первым параметром входного сообщения метода, инициирующего выполнение сервиса, должен быть идентификатор экземпляра ПЭВ, обращающегося к сервису. Единственным параметром выходного сообщения – результат запуска сервиса (сервис

успешно запущен (RESULT_OK), внутренняя ошибка метода (RESULT_FAILED), экземпляр ПЭВ не имеет прав на обращение к сервису (RESULT_ACCESS_DENIED)).

```
<message name="RPCServiceStartRequest">
  <part name="eip_id" type="string"/>
   ...
</message>

<message name="RPCServiceStartResponse">
  <part name="result" type="string"/>
</message>

<operation name="RPCServiceStart">
  <input message="RPCServiceStartRequest"/>
  <output message="RPCServiceStartResponse"/>
</operation>
```

Единственным параметром входного сообщения метода, обеспечивающего получение значений выходных параметров сервиса, является идентификатор ПЭВ, обратившегося к сервису. Первым параметром выходного сообщения такого метода должен быть статус завершения работы метода (успешное завершение (RESULT_OK), выполнение сервиса еще не завершено (RESULT_NOT_DONE), внутренняя ошибка метода (RESULT_FAILED)).

```
<message name="RPCServiceResultsRequest">
  <part name="eip_id" type="string"/>
</message>

<message name="RPCServiceResultsResponse">
  <part name="result" type="string"/>
   ...
</message>

<operation name="RPCServiceResults">
  <input message="RPCServiceResultsRequest"/>
```

```
    <output message="RPCServiceResultsResponse"/>
  </operation>
```

При обращении некоторого экземпляра ПЭВ к web-сервису последний должен проверить, обладает ли данный ПЭВ правами на его запуск.

Для этого разрабатываемый субъектом web-сервис должен обладать операцией, позволяющей осуществить обращение к подсистеме безопасности Среды для проверки наличия у обратившегося к сервису экземпляра ПЭВ прав на выполнение сервиса. Параметрами входного сообщения такой операции являются идентификаторы самого web-сервиса и обратившегося к нему экземпляра ПЭВ; единственным параметром выходного сообщения – результат проверки прав доступа (доступ разрешен (ACCESS_GRANTED) или запрещен (ACCESS_DENIED)).

```
<message name="CheckAccessSolicit">
  <part name="eip_id" type="string"/>
  <part name="ws_id" type="string"/>
</message>

<message name="CheckAccessResponse">
  <part name="result" type="string"/>
</message>

<operation name="CheckAccess">
   <output message="CheckAccessSolicit"/>
   <input message="CheckAccessResponse"/>
</operation>
```

Разрабатываемый субъектом web-сервис должен обладать методом, позволяющим получить информацию о состоянии выполнения сервиса в рамках некоторого экземпляра ПЭВ. Единственным параметром входного сообщения этого метода является идентификатор обратившегося к сервису экземпляра ПЭВ; единственным параметром выходного – статус выполнения web-сервиса (выполняется (STATUS_PROCESSING), приостановлен (STATUS_SUSPENDED), ожидает ввода данных (STATUS_WAITING_FOR_DATA), находится в процессе остановки (STATUS_PROCESSING_STOP) или отката (STATUS_PROCESSING_ROLLBACK),

обращение указанного экземпляра ПЭВ уже завершено (STATUS_DONE) или не происходило (STATUS_NOT_CALLED), внутренняя ошибка метода (STATUS_FAILED)).

```
<message name="StatusRequest">
   <part name="eip_id" type="string"/>
</message>

<message name="StatusResponse">
   <part name="status" type="string"/>
</message>

<operation name="GetStatus">
    <input message="StatusRequest"/>
    <output message="StatusResponse"/>
</operation>
```

Разрабатываемый субъектом web-сервис должен обладать методом, позволяющим получить детальную информацию о выполнении бизнес-процесса сервиса в рамках некоторого экземпляра ПЭВ. Единственным параметром входного сообщения этого метода является идентификатор обратившегося к сервису экземпляра ПЭВ; параметрами выходного являются строка HTML-кода, описывающая состояние бизнес-процесса (с учетом необходимого форматирования и внедренными ссылками на описываемые документы), и результат выполнения метода (запрашиваемое описание предоставлено (RESULT_OK), дополнительная информация в настоящий момент не может быть предоставлена (RESULT_NO_BUSINESS_CONTEXT), обращение указанного экземпляра ПЭВ уже завершено (RESULT_DONE) или не происходило (RESULT_NOT_CALLED)).

```
<message name="BusinessContextRequest">
   <part name="eip_id" type="string"/>
</message>

<message name="BusinessContextResponse">
   <part name="description" type="string"/>
   <part name="result" type="string"/>
</message>
```

```
<operation name="GetBusinessContext">
    <input message="BusinessContextRequest"/>
    <output message="BusinessContextResponse"/>
</operation>
```

В процессе выполнения web-сервиса должен происходить обмен данными с пользователем посредством некоторого пользовательского web-интерфейса. Для этого в любой момент времени web-сервис должен обладать возможностью передать в Среду ссылку на web-интерфейс, предназначенный для получения от пользователя необходимой для выполнения сервиса информации. После того, как указанные данные получены web-сервисом, он продолжает выполнение. В случае, если на момент обращения сервис не нуждается в получении дополнительной информации, он возвращает значение результата «дополнительная информация не требуется» (RESULT_NOT_WAITING_FOR_DATA) и пустую строку в качестве ссылки. Для осуществления возможности указанного интерактивного взаимодействия разрабатываемый субъектом web-сервис должен обладать методом, возвращающим указанную ссылку. Единственным параметром входного сообщения такого метода является идентификатор обратившегося к сервису экземпляра ПЭВ; параметрами выходного – ссылка на указанный интерфейс и результат обращения (запрашиваемая ссылка предоставлена (RESULT_OK), дополнительная информация не требуется (RESULT_NOT_WAITING_FOR_DATA), обращение указанного экземпляра ПЭВ уже завершено (RESULT_DONE) или не происходило (RESULT_NOT_CALLED)).

```
<message name="UserInteractRequest">
   <part name="eip_id" type="string"/>
</message>

<message name="UserInteractResponse">
   <part name="result" type="string"/>
   <part name="url" type="string"/>
</message>

<operation name="UserInteract">
    <input message="UserInteractRequest"/>
    <output message="UserInteractResponse"/>
</operation>
```

Разрабатываемый субъектом web-сервис должен обладать методом, позволяющим остановить выполнение сервиса в рамках некоторого экземпляра ПЭВ. Единственным параметром входного сообщения этого метода является идентификатор обратившегося к сервису экземпляра ПЭВ; единственным параметром выходного – результат попытки останова сервиса (выполнение сервиса остановлено (RESULT_OK), выполнение сервиса не может быть остановлено (RESULT_FAILED), сервис уже находится в процессе остановки (RESULT_PROCESSING_STOP), обращение указанного экземпляра ПЭВ уже завершено (RESULT _DONE) или не происходило (RESULT _NOT_CALLED)).

```
<message name="StopRequest">
   <part name="eip_id" type="string"/>
</message>

<message name="StopResponse">
   <part name="result" type="string"/>
</message>

<operation name="Stop">
    <input message="StopRequest"/>
    <output message="StopResponse"/>
</operation>
```

7.1.5.2Дополнительные требования к web-сервисам

Для большинства сервисов в любой момент до завершения выполнения web-сервиса в рамках некоторого экземпляра ПЭВ может быть произведен откат выполнения сервиса, т.е. сервис в рамках ПЭВ может быть приведен в исходное состояние.

Разрабатываемый субъектом web-сервис должен обладать методом, позволяющим произвести откат выполнения сервиса в рамках некоторого экземпляра ПЭВ к его первоначальному состоянию. Единственным параметром входного сообщения этого метода является идентификатор обратившегося к сервису экземпляра ПЭВ; единственным параметром выходного – результат попытки отката сервиса (откат сервиса произведен, откат сервиса не может быть произведен, сервис уже находится в процессе отката, обращение указанного экземпляра ПЭВ уже завершено и поэтому откат невозможен, обращение указанного экземпляра ПЭВ не происходило). Сервисы, откат которых непредусмотрен, должны обладать указанным методом, но всегда возвращать результат «откат сервиса не может быть произведен».

```
<message name="RollbackRequest">
<part name="eip_id" type="string"/>
</message>

<message name="RollbackResponse">
<part name="result" type="string"/>
</message>

<operation name="RollbackWS">
    <input message="RollbackRequest"/>
    <output message="RollbackResponse"/>
</operation>
```

В процессе развития программного комплекса Среды электронного взаимодействия и наращивания его функциональности требуется производить тестирование системы в реальном времени. Для осуществления такого тестирования разрабатываемый субъектом web-сервис для каждого метода, реализующего его функционал, должен обладать парным проверочным методом, позволяющим произвести проверку функциональности основного метода, не затрачивая реальные время и ресурсы на выполнение всех его действий. Параметры входного сообщения проверочного метода совпадают с параметрами входного сообщения основного метода, а выходное сообщение проверочного метода содержит все параметры выходного сообщения и результат тестирования сервиса (метод выполнен без ошибок, во время тестирования происходили внутренние ошибки сервиса, результат выполнения метода недостоверен, экземпляр ПЭВ не имеет прав на обращение к сервису).

```
<message name="ServiceTestRequest">
   <part name="eip_id" type="string"/>
    ...
</message>

<message name="ServiceTestResponse">
   <part name="result" type="string"/>
    ...
   <part name="testresult" type="string"/>
</message>
```

```
<operation name="ServiceTest">
    <input message="ServiceTestRequest"/>
    <output message="ServiceTestResponse"/>
</operation>
```

7.2 Требования к интегрируемой ИС

Интегрируемая в СЭВ информационная система должна удовлетворять:

- требованиям к поддерживаемым СЭВ стандартам и рекомендациям по их применению в рамках СЭВ, описанным в разделе «6. Рекомендации по использованию стандартов в рамках СЭВ»;
- требованиям к оформлению адаптеров интегрируемой системы (внутренних WEB-сервисов СЭВ), описанным в разделе «7.1.5 Требования к внутренним web-сервисам СЭВ».

Помимо этого интегрируемая информационная система обязана:

- поддерживать авторизацию пользователей средствами модуля авторизации СЭВ, описанными в разделе «7.1.4.1 Описание сервисов модуля авторизации»;
- регистрировать все предоставляемые сервисы средствами модуля UDDI реестра СЭВ, описанными в разделе «7.1.4.2 Описание сервисов модуля UDDI»;
- для классификации своих услуг пользоваться стандартными классификаторами СЭВ, доступ к которым осуществляется через интерфейсы подсистемы классификаторов, описанные в пункте «7.1.4.4 Описание сервисов модуля классификаторов».

7.3 Организация работ по проведению интеграции ИС с СЭВ

Для интеграции информационной системы в СЭВ необходимо провести следующий состав работ.

7.3.1 Этап 1. Описание данных ИС в терминах стандартных схем данных СЭВ

На данном этапе необходимо выполнить отображение существующей схемы данных информационной системы на объектные стандартные схемы ресурсов СЭВ для

возможности информационного взаимодействия. Результатом работ по данному этапу должны являться правила сопоставления элементов и атрибутов XML-описания ресурсов ИС и XML-описания стандартных сущностей ИС (канонической схемы данных). Работы по данному этапу разбиваются на следующую последовательность действий:

7.3.1.1 Выделение базовых (общих) структур данных

В рамках данных работ необходимо провести анализ существующих данных, размещаемых в хранилище данных ИС. После этого необходимо провести выделение объектной схемы ресурсов ИС на базе проведенного анализа. Результаты преобразования существующей модели данных к объектной необходимо представить в виде набора XML-схем, описывающих информационные сущности ИС.

7.3.1.2 Построение преобразователей от общих к каноническим схемам данных

В рамках данных работ необходимо создать XSLT-преобразователи общих типов данных ИС к соответствующим им типам данных (определенным на предыдущем этапе) канонических (стандартных) схем СЭВ. При этом необходимо обеспечить двусторонне преобразование для того, чтобы было возможным не только приведение формата данных ИС к формату данных, воспринимаемому СЭВ, но и обратного приведения структур данных СЭВ к структурам конкретной ИС (средства автоматизации построения адаптера). Далее схемы данных ИС и их преобразователи к каноническим схемам необходимо зарегистрировать в репозитарии схем данных СЭВ. Результатом данного этапа являются зарегистрированные в СЭВ средства прямой и обратной трансформации данных ИС к формату данных СЭВ.

7.3.1.3 Требования к квалификации сотрудников и приблизительные трудозатраты

Для выполнения указанных работ требуются два специалиста. Первый из них – эксперт – должен обладать хорошим знанием предметных области интегрируемой ИС, достаточными для выделения общих и канонических структур данных на вербальном уровне. Второй специалист должен обладать хорошими знаниями стандарта W3C XML Schema и владеть технологией XSLT для формирования преобразователей данных.

Приблизительные трудозатраты на весь процесс приведения форматов данных ИС к форматам данных СЭВ – 20 чел/дней для каждого из участников процесса.

7.3.2Этап 2. Реализация адаптера ИС (WEB-сервиса) для взаимодействия с сервисами СЭВ в рамках стандартных схем данных СЭВ

На этом этапе происходит построение адаптера ИС (WEB-сервиса), позволяющего выполнять обмен данными ИС и СЭВ, и использующего средства трансформации данных, определенные на предыдущем этапе. Помимо интерфейса обмена данными, данный WEB-сервис может предоставлять дополнительные операции по манипулированию данными (поиск, загрузка, выгрузка, создание, модификация, удаление и т.д.). Интерфейсы WEB-сервиса адаптера должны соответствовать требованиям, описанным в разделе «7.1.5 Требования к внутренним web-сервисам СЭВ». Результатом данного этапа является созданный адаптер ИС, поддерживающий стандартный интерфейс и позволяющий осуществить интеграцию с СЭВ. Работы по данному этапу разбиваются на следующую последовательность действий:

7.3.2.1Поддержка требований к внутренним сервисам СЭВ

В рамках данных работ необходимо реализовать Web-сервис для взаимодействия ИМ с СЭВ с набором специфичных для задач конкретной ИС операций. Необходимо привести интерфейс данного сервиса к интерфейсу, удовлетворяющему требованиям к оформлению интерфейсов внутренних сервисов СЭВ, описанным в разделе «7.2 Требования к интегрируемой ИС». Данные доработки предполагают расширение WSDL-интерфейса адаптера ИС дополнительными операциями, используемыми для взаимодействия с сервисом исполнения ПЭВ СЭВ.

7.3.2.2Поддержка стандартов взаимодействия СЭВ

Для взаимодействия с сервисом исполнения ПЭВ СЭВ Web-сервис ИС должен следовать следующим требованиям к транспортировке сообщений:

- поддерживать привязку SOAP-сообщений к транспортным протоколам HTTP/HTTPS с методом POST-привязки в соответствии с рекомендациями, описанными в разделе «5.2.2 Описание способов инкапсуляции SOAP-сообщения в различные транспортные протоколы»;
- поддерживать стандартный RPC-стиль оформления SOAP-сообщений;
- все структуры данных в формате W3C XMLSchema хранить отдельно (в репозитарии схем данных СЭВ) и осуществлять их импорт по URI в WSDL-интерфейсе Web-сервиса ИС.

Для привязки SOAP-сообщений к протоколам HTTP/HTTPS в рамках СЭВ будет использоваться стандартная привязка посредством HTTP-метода POST для передачи запроса и ответа сервиса в теле POST-запроса. Использование SOAP-сообщений дано в спецификациях по SOAP.

Для безоапасного и надежного функционирования в рамках СЭВ Web-сервис ИС должен:

- пользоваться сервисом авторизации СЭВ, интерфейсы которого описаны в разделе «8.1.4.1 Описание сервисов модуля авторизации», для аутентификации и авторизации пользователей СЭВ, вызывающих его операции. WSDL-интерфейс сервиса авторизации СЭВ описан в приложении D в разделе «13.4.1 WSDL-определение интерфейса сервиса авторизации»;
- поддерживать требования транзакционности на уровне дополнительных операций интерфейса для поддержки транзакционных протоколов СЭВ, описанные в разделе «8.1.5.2 Дополнительные требования к web-сервисам».

Для преобразования данных между форматами данных СЭВ и локальными форматами данных Web-сервис СЭВ должен пользоваться средствами, предоставляемыми репозитарием схем данных СЭВ, программные интерфейсы доступа к которому описаны в разделе «8.1.4.5 Описание модуля репозитарий». WSDL-интерфейс сервиса репозитария СЭВ описан в приложении D в разделе «13.4.5 WSDL-определение интерфейса репозитария».

7.3.2.3Регистрация адаптера ИС в UDDI реестре СЭВ

В рамках данных работ осуществляется регистрация созданного на предыдущем шаге адаптера ИС в UDDI реестре СЭВ для возможности поиска и использования предоставляемых им услуг при проектировании интегрирующих ПЭВ. При этом необходимо использовать следующие средства, предоставляемые СЭВ:

- Для классификации регистрируемого адаптера внешней ИС необходимо воспользоваться унифицированными классификаторами, расположенными в подсистеме классификаторов СЭВ (раздел «8.1.4.4 Описание сервисов модуля классификаторов»), доступ к которой можно получить с помощью пользовательских интерфейсов подсистемы ССК;
- Для регистрации адаптера в UDDI репозитарии СЭВ необходимо либо воспользоваться пользовательскими интерфейсами модуля UDDI СЭВ, описанного

в разделе «8.1.4.2 Описание сервисов модуля UDDI», либо программными интерфейсами сервиса UDDI.

Для регистрации адаптеров ИС ЭАР в UDDI реестре СЭВ предусмотрен GUI-интерфейс пользователя интегрированный в портал СЭВ.

Интерфейс пользователя предоставляет возможность простого и расширенного поиска организаций и услуг, интерфейс для публикации данных об организациях и услугах и выборку данных, опубликованных пользователем. В экраны пользовательского интерфейса интегрированы также GUI ССК (выбор классификатора и значений классификатора). При просмотре результатов поиска предоставляется регистрационная информация об услугах и организациях, а так же, в случае, если услуга является ПЭВ, ссылка на страницу запуска ПЭВ.

Кроме того в СЭВ предусмотрен GUI-интерфейс администратора, который предоставляет возможность добавления, удаления и изменения опубликованных данных, доступ к настройке пользовательского интерфейса и настройкам соединения с Oracle UDDI (имя, пароль, *url* и т.п.).

Результатом данных работ является полноценная интеграция ИС в СЭВ с поддержкой динамического доступа к интерфейсам адаптера ИС.

7.3.2.4 Требования к квалификации сотрудников и приблизительные трудозатраты

Для выполнения указанных работ требуются два специалиста. Первый из них – эксперт – должен иметь хорошее знание предметной области ИС для привязки их к системе классификации услуг UDDI реестра СЭВ. Второй специалист должен обладать:

- хорошими знаниями стандартов W3C SOAP, W3C WSDL;
- умением разрабатывать Web-сервисы на платформе интегрируемой ИС.

Приблизительные трудозатраты на весь процесс реализации адаптера – 20 чел/дней технического специалиста и 5 чел/дней эксперта.

8 Рекомендации по использованию стандартов в рамках ИС ЭАР.

В данном разделе освещается специфика использования данных стандартов в рамках ИС ЭАР. Прежде всего, указывается поддерживается ли и в каком объеме та или иная спецификация, указываются характерные особенности и ограничения по ее применению. При этом данные ограничения могут быть обусловлены как архитектурой СЭВ, так и текущей реализацией интеграционных механизмов.

8.1 ИС и сервисы

ИС ЭАР имеет встроенную поддержку Web-сервисов. ЭАР может использовать внешние веб-сервисы с подсистемы взаимодействия, равно как и операции самого ЭАР могут быть выставлены в виде Web-сервисов.

ИС ЭАР использует и полностью поддерживает язык WSDL 1.1 для описания Web-сервисов, что соответствует общим рекомендациям стандартов для СУВ.

ИС ЭАР не накладывает дополнительных требований на использование стандарта WSDL 1.0 для описания интерфейсов Web-сервисов среды помимо тех рекомендаций, которые описаны в пункте «5.1 ИС и сервисы». При этом не вносится дополнительных расширений в спецификацию.

8.2 Транспорт и сообщения

ИС ЭАР использует и полностью поддерживает протокол SOAP 1.1 как транспортный уровень Web-сервисов, что соответствует общим рекомендациям стандартов для СУВ.

ИС ЭАР не накладывает дополнительных требований на использование стандарта SOAP 1.0 для обмена XML-сообщениями помимо тех рекомендаций, которые описаны в пункте «5.2 Транспорт и сообщения». При этом не вносится дополнительных расширений в спецификацию.

В работах 2004-го года предполагается использовать только протоколы HTTP/HTTPS для транспорта сообщений SOAP. Такое решение было принято вследствие того, что данные протоколы позволяют решить все необходимые задачи и при этом

достаточно просты, и широко используемы. Также раздел 5.2 и WS-I Basic Profile 1.0 рекомендуют использование именно HTTP/HTTPS для транспорта сообщений SOAP.

8.3 Каноническая модель данных и язык спецификации схем данных

8.3.1 Рекомендации по использованию единой описательной модели для интеграции данных

Использование технологии обмена с разрабатываемыми АИС информацией на базе XML является необходимым условием функционирования ИСЭАР.

ИС ЭАР ориентируется на парадигму сервис-ориентированной архитектуры (SOA), в которой сообщения и документы описываются на XML. Деловые процессы, ЭАРы, внутренне работают только с XML-документами. Для описания схем XML-документов используется стандарт XML Schema. Для создания и редактирования XML-схем ИСЭАР включает редактор XML-схем. Схемы могут быть также импортированы из внешних источников.

Итого, ИС ЭАР использует и полностью поддерживает язык XML как единый формат обмена и язык XML Schema для описания схем XML-документов, что соответствует общим рекомендациям стандартов для СУВ.

Следующая сводная таблица указывает поддерживаемые в ИС ЭАР средства интеграции данных, опирающихся на XML как единую каноническую модель данных.

Таблица 5. Сводная таблица, поддерживаемых ИС ЭАР средств интеграции данных

Средство интеграции	Какими средствами предоставляется
Проверка правильности данных	XML-схемы
Определение схем	редактор XML-схем
Преобразование данных	XSLT, редактор отображения XML-схем
Распознавание схем	XML-схемы

Конкретные базовые XML-схемы, используемые в ИС ЭАР, описаны в разделе 7.10 «Описание базовых схем данных для сквозных пилотных проектов ИС ЭАР» и 7.11 «Контролируемые словари в НСИ ИС ЭАР».

8.3.2 Рекомендации по использованию стандарта W3C XML Schema в рамках ИСЭАР для описания модели данных

8.3.2.1 Специфика использования стандарта

ИС ЭАР не накладывает дополнительных требований на использование стандарта XML Schema для описания схем XML-документов, рекомендуется придерживаться указаний, которые описаны в пункте «5.3 Каноническая модель данных и язык спецификации схем данных». При этом не вносится дополнительных расширений в спецификацию.

8.3.2.2 Способ организации схем данных

Схемы данных ИСЭАР поддерживают модульность организации схем посредством стандартных механизмов импорта, определяемых стандартом W3C XML Schema.

8.3.2.3 Средства проектирования схем данных

При проектировании схем данных СЭВ были использованы методика и средства UML-моделирования, описанные в разделе «5.3.1 Руководство по процессу разработки XML-схем». Диаграммы можно увидеть в соответствующих разделах 7.10 «Описание базовых схем данных для сквозных пилотных проектов ИС ЭАР» и 7.11 «Контролируемые словари в НСИ ИС ЭАР».

8.4 Регистрация сервисов и метаинформации

Подсистема описания ИСЭАР предназначена для хранения следующих описаний и метаданных:

- Метаданные АИС, разрабатываемых в рамках проекта ЭАР;
- схем документов (XML-схемы);
- типов сообщений;
- электронных административных регламентов.

Состав подсистемы описания ИСЭАР рассмотрен в разделе 9.1.1.3 «Состав ИСЭАР».

В рамках работ 2004 года не планируется использовать UDDI-реестр для каталогизации Web-сервисов, такая потребность может возникнуть в рамках работ следующих этапов.

На данном этапе ИСЭАР не накладывает дополнительных требований и не выдвигает никаких рекомендаций по регистрации сервисов и метаинформации помимо

общих рекомендаций, рассмотренных в разделе 5.4 «Регистрация сервисов и метаинформации».

8.5 Безопасность

Задачи безопасности в ИСЭАР в рамках данного этапа предполагается решать через два основных механизма: протокол HTTPS для транспорта в сети Интернет и авторизацию подсистемы взаимодействия. Протокол HTTPS был выбран как наиболее простой и надежный механизм, который на данный момент широко используется во многих системах. Раздел 5.5 и WS-I Basic Profile 1.0 рекомендуют использование HTTPS для задач безопасности.

На данном этапе ИС ЭАР не поддерживает стандарт WS-Security. Поддержка этого и других связанных стандартов может быть добавлена на более поздних этапах.

8.6 Транзакции

Подсистема описания деловых процессов ИС ЭАР позволяет указывать в процессах такие конструкции, как:

- область видимости – позволяет организовать транзакции и обработку исключительных ситуаций
- компенсация - позволяет определить действия для отмены (компенсации) уже произведенных действий в сценарии

Подсистема исполнения деловых процессов ИС ЭАР поддерживает как короткие, атомарные (ACID) транзакции, так и долгосрочные транзакции. Все типы транзакций допускают указание логики компенсации, которая позволяет отменить результаты исполнения транзакции или предоставляет альтернативную серию действий обработчика ошибки, которые надо предпринять для отката изменений.

На данном этапе ИС ЭАР не поддерживает стандарт WS-Transactions. Поддержка стандартов транзакционности Web-сервисов может быть добавлена на более поздних этапах.

На данном же этапе для реализации транзакционности в ЭАР выбрана модель бизнес-транзакций (LRT). Транзакционность в деловых процессах обеспечивается средствами самого языка описания деловых процессов подсистемы описания деловых процессов ИС ЭАР (присутствующими также в BPEL4WS) на базе средств обработки

исключительных ситуаций и компенсации. См. общие рекомендации раздела 5.6, в частности раздел 5.6.2 «Рекомендации по реализации координирующих протоколов бизнес-транзакций на базе средств обработки исключительных ситуаций и компенсации языка описания АРП BPEL4WS»

8.7 Рабочие процессы

Подсистема описания и подсистема исполнения деловых процессов ИС ЭАР предоставляет возможность визуальной разработки и исполнения деловых процессов, поддерживающих исчерпывающий набор программных конструкций, таких как транзакции и обработка исключений, поддержка долгосрочных процессов, управление состоянием процесса. Детали этих подсистем описаны в разделах 9.1.3.1.1. «Подсистема описания деловых процессов ИС ЭАР» и 9.1.3.1.2 «Подсистема исполнения деловых процессов ИС ЭАР».

Деловой процесс, описанный средствами ИС ЭАР, может быть экспортирован в формат BPEL4WS версии 1.1 для интероперабельного обмена описаниями рабочих процессов среди СУВ. Подсистема описания деловых процессов ИС ЭАР поддерживает также импорт описаний процессов на BPEL4WS. Детали импорта и экспорта BPEL4WS-описаний раскрыты в разделах 7.7.1 и 7.7.2

Фактически, объекты-примитивы моделирования делового процесса в подсистеме описания деловых процессов ИС ЭАР являются прямым представлением BPEL4WS-элементов, таких как получение сообщения, вызов Web-сервиса, последовательность действий, поток действий, условные конструкции, роли, ссылки и источники.

Следует, однако, учитывать, что BPEL4WS ориентирован исключительно на Web-сервисы, тогда как помимо поддержки этого языка подсистема исполнения деловых процессов ИС ЭАР предоставляет более широкий набор сервисов. Например, предоставляется возможность преобразования XML-данных от одной XML-схемы к другой, возможность вызова методов локальных объектов, поддержка транзакций и пр. элементы, недоступные в BPEL4WS.

8.7.1Импорт BPEL4WS в ИС ЭАР

В реализации 2004 года описание деловых процессов в ИС ЭАР базируется на Microsoft BizTalk Server 2004, и соответственно на языке XLANG/s. Соответственно, импорт BPEL4WS 1.1 в ИС ЭАР имеет следующие ограничения, имеющиеся в BizTalk Server 2004:

- При импорте BPEL4WS и WSDL, имя вершины WSDL-определения и вершины BPEL4WS-процесса не должны совпадать.
- В импортируемом BPEL4WS не допускается использование зарезервированных выражений (в частности, см. список зарезервированных выражений XLANG/s [30] в квадратных скобках)
- Поддерживаются только предопределённые простые XSD-типы
- XSD-тип xsd:QName не поддерживается; вместо него следует использовать xsd:string.

8.7.2Экспорт BPEL4WS из ИС ЭАР

При экспорте в BPEL4WS требуется, чтобы ЭАР содержал только общие конструкции между языком подсистемы описания деловых процессов ИС ЭАР и языком BPEL4WS, или конструкции, которые могут быть транслированы в BPEL4WS без изменения поведения процесса. Перечисленные ниже конструкции недопустимы при экспорте в BPEL4WS:

- Не может быть использован компонент «Вызов» или компонент «Старт».
- Не может быть использован компонент «Трансформация»
- Нельзя указать таймаут долгосрочной транзакции.
- Деловой процесс не может принимать параметры.
- Обработчики компенсаций не могут принимать параметры.
- Типы переменных должны быть совместимы с типами в XPath.
- Недопустимо применение компонента «Приостановить».
- Тип литерала может быть одним из: boolean, char, byte, sbyte, int32, uint32, int64, uint64, single, double, string
- В арифметических операциях допустимы только операнды следующих типов: byte, sbyte, int32, uint32, int64, uint64, single, double
- Сравнительные операторы не могут использоваться с типом char.
- В выражениях нельзя указывать ссылки на servicelink.

- Между вызовами компонент «Отправка» и «Получение» не могут быть указаны какие-либо действия, если они используют общий порт отправки-получения.

Замечание. Литералы-символы экспортируются как беззнаковые целые числа.

Замечание. Названия идентификаторов должны удовлетворять требованиям спецификации Extensible Markup Language (XML) 1.0.

Замечание. Если процесс содержит присвоение ролевой ссылки или сервисной ссылки (servicelink), то для конца ссылки генерируется BPEL4WS-заглушка и выводится предупреждение.

8.8 Глобальная идентификация

В рамках работ 2004 года для глобальной идентификации информационных ресурсов, с которыми работают ЭАР, и идентификации информационных ресурсов НСИ ИС ЭАР используется URI-идентификация на основе GUID. Сервис идентификации ИС ЭАР позволяет АИС резервировать URI-идентификаторы для описания своих информационных ресурсов.

Уточнение требований по глобальной идентификации может быть проведено в рамках работ следующих этапов.

8.9 Глоссарии

8.9.1 Глоссарий ИС ЭАР

Деловой процесс

Логическая последовательность взаимозависимых операций (действий), которые используют ресурсы для создания или получения в обозримом или предсказуемом будущем измеримого результата.

Функция государственного органа

Нормативно закрепленный круг полномочий и обязанностей государственного органа, реализующийся через достижение целей и выполнение задач этим органом.

Управление

Элемент, функция государственной системы, обеспечивающая сохранение ее определенной структуры, поддержание режима деятельности, реализацию программ.

Регламент

Ссовокупность правил, определяющих порядок деятельности.

Административный регламент (АР)

Комплексное описание порядка выполнения отдельного делового процесса, обеспечивающего реализацию функций органа исполнительной власти. Административный регламент включает организационные и технологические элементы и содержится в документе, правовой статус которого определяется соответствующим государственным органом.

Электронный административный регламент (ЭАР)

Нормативно закрепленный регламент выполнения деловых процессов с обязательным использованием информационных технологий.

Автоматизированная система (АС)

Система, состоящая из персонала и комплекса средств автоматизации его деятельности, реализующая информационную поддержку выполнения установленных функций.

Анализ «затраты-выгоды»

Метод, используемый для сравнения различных затрат, связанных с инвестициями, с прибылью, которую предполагается получить. В анализе «затраты-выгоды» предполагается рассматривать и учитывать материальные и нематериальные факторы.

Артефакт

Абстрактное представление некоторого аспекта существующей или проектируемой системы, компонента или представления. Примерами артефактов являются графическая модель, структурная модель, табличные данные, структурированные или неструктурированные комментарии. Отдельные артефакты могут объединяться (агрегироваться).

Архитектура

Описание (модель) базовой компоновки и взаимосвязей частей системы (физического либо концептуального объекта или сущности). Замечание: Существует два и только два типа архитектур, имеющих дело с интеграцией предприятий:

а) Системные архитектуры (иногда называемые архитектурами «типа 1», которые имеют дело с конструкцией системы, т.е. с частью компьютерной системы управления общей системы интеграции предприятия;

б) Проекты эталона предприятия (иногда называемые архитектурами «типа 2», которые имеют дело с организацией разработки и реализации такого проекта, как интеграция предприятия или других программ развития предприятий.

Архитектура «как должно быть»

Целевое состояние архитектуры предприятия (см. также Целевая архитектура).

Архитектура «как есть»

См. Базовая архитектура

Архитектура географического развертывания системы

Логическая модель реализации системы материально-технического обеспечения бизнеса предприятия. Она характеризует типы системного оборудования и управляющего программного обеспечения на узлах и линиях (например, процессоры и операционные системы, устройства памяти, СУБД, периферийные устройства и драйверы, коммуникационное программное обеспечение и т.д.).

Архитектура данных

Один из компонентов технической архитектуры проекта. Архитектура данных состоит, наряду с другими категориями, из объектов данных, которые имеют атрибуты и связи с другими объектами данных. Все объекты, отражаемые в архитектуре данных, связаны с бизнес-функциями предприятия.

Архитектура предприятия

Стратегический информационный базовый ресурс данного конкретного предприятия, который определяет бизнес предприятия, информацию, необходимую для его осуществления, технологии, необходимые для поддержки бизнес-операций, а также переходные процессы, необходимые для реализации новых технологий, соответствующих изменяющимся потребностям бизнеса. Архитектура предприятия может иметь вид четко представленного документа либо наметок.

Архитектура приложений

Компонент технической архитектуры предприятия, который определяет основные приложения, необходимые для управления данными и для поддержки бизнес-функций.

Архитектурная политика предприятия

Руководящие принципы разработки, реализации и поддержки архитектуры предприятия.

Архитектурный артефакт

Релевантная документация, модели, диаграммы, описания и результаты анализа, а также базовый репозиторий, стандарты и профили защиты.

Архитектурный продукт

Структура компонентов, их взаимосвязи, принципы и рекомендации по управлению их проектированием и последующим развитием.

Архитектурный продукт предприятия

Графические материалы, модели и/или вербальные описания, которые представляют среду и конструкцию предприятия.

Архитектурный репозиторий

См. Репозиторий архитектуры

Архитектурный сегмент

См. Сегмент архитектуры

Атрибут

Порция информации, определяющая некоторое свойство сущности.

Аудит

Проверка, выполняемая компетентным органом (лицом) с целью обеспечения независимой оценки степени соответствия программных продуктов или процессов установленным требованиям.

Базис проекта

Основополагающие параметры и фиксирующие их согласованное понимание всеми участниками документы проекта - «точка опоры» для всего последующего развития проекта.

Базовая архитектура

1) Совокупность продуктов, которые представляют существующее предприятие, существующую практику бизнеса и техническую инфраструктуру. 2) Существующее состояние архитектуры

Базовая линия

Официально принятая версия элемента конфигурации, независимая от среды, формально обозначенная и зафиксированная в конкретный момент времени жизненного цикла элемента конфигурации.

Базовый план

Первоначальный план проекта с утвержденными изменениями. Базовый план бывает также и по составляющим проекта - стоимости, расписанию и т. д.

Безопасность системы

1) Защищенность системы от несанкционированного использования ее функциональных возможностей, а также от возможных нарушений ее функционирования, вызванных различными предсказуемыми и непредсказуемыми обстоятельствами. 2) Безопасность системы - это не только ее характеристика, но и набор механизмов, охватывающих систему как логически, так и физически.

Библиотека инфраструктуры информационных технологий

Инициированный под эгидой Министерства Великобритании, а впоследствии международный проект по формированию коллекции передовых практических рекомендаций и руководств по организации связанных с ИТ бизнес-процессов. Коллекция

поддерживается Центральным компьютерным и телекоммуникационным агентством (The Central Computer and Telecommunication Agency, CCTA, Лондон, 1989).

Бизнес-архитектура

Компонент текущей и целевой архитектуры, относящийся к основной федеральной задаче (миссии) данного предприятия и соответствующий его целям. Бизнес-архитектура включает содержание бизнес-моделей и концентрируется на федеральных областях бизнеса и бизнес-процессах, соответствующих бизнес-стимулам. Бизнес-архитектура определяет федеральные бизнес-процессы, федеральные информационные потоки, а также информацию, необходимую для осуществления бизнес-функций предприятия.

Бизнес-модель

Компонент моделей архитектуры, представляющий существующую и целевую федеральные архитектуры бизнеса. Бизнес-модели характеризуют бизнес-данные, которые используются для определения деловых потребностей, процессов и информации.

Бизнес-процесс

1) Частично упорядоченное множество деятельностей предприятия, которые могут выполняться для того, чтобы достигнуть данной цели предприятия или части предприятия, обеспечивающей получение желаемого конечного результата. 2) Совокупность взаимосвязанных структурированных деятельностей - цепочка событий, которая продуцирует некоторую конкретную услугу или продукт для конкретного потребителя или потребителей.

Верификация

Подтверждение экспертизой и представлением объективных доказательств того, что конкретные требования полностью реализованы.

Внешняя среда

Неконтролируемая часть системы, которая распространяется в такой степени, что процедура принятия решений для управления такой системой не может быть постигнута.

Двигатель архитектуры

В соответстсвии с Руководством по разработке федеральной архитектуры предприятия, факторы, которые обозначают внешние стимулы, вызывающие необходимость изменения архитектуры предприятия. Двигатели архитектуры делятся на категории: бизнес-двигатели и технические двигатели.

Деловая архитектура

См. Бизнес-архитектура

Деловой процесс

См. Бизнес-процесс

Деятельность

Полная функциональность или ее часть. Замечание: Деятельность предприятия состоит из элементарных задач, выполняемых предприятием, которое потребляет входы и распределяет время и ресурсы для продуцирования выходов.

Доступность <информации>

Свойство информации, заключающееся в возможности доступа к ней бизнес-процесса, когда она ему требуется в настоящее время и в будущем.

Жизненный цикл разработки системы

1) Конечное множество родовых фаз и шагов, которые система может проходить на протяжении полной истории ее жизни. 2) Руководство, стратегии и процедуры, предназначенные для разработки системы на протяжении всего ее жизненного цикла, включая определение требований, проектирование, реализацию, проведение испытаний, развертывание, функционирование и техническое обслуживание. 3) Сфера деятельности, связанная с некоторой системой, охватывающая инициацию проекта системы, ее разработку и комплектование, реализацию, функционирование и техническое обслуживание и, в конечном счете, избавление от нее, вызванное инициацией проекта другой системы.

Жизненный цикл системы

См. Жизненный цикл разработки системы.

Заказчик

1) Организация, которая приобретает или получает систему, программный продукт или программную услугу от поставщика. 2) Индивидуум или организация, которым передается данный продукт или оказывается услуга. Заказчик может также быть конечным пользователем.

Затраты

1) Термин, используемый для обозначения расхода денежных средств на конкретную альтернативу инвестиций в течение ожидаемого периода времени. Затраты могут включать прямые и косвенные начальные затраты, любые периодические и непрерывные затраты на некоторую операцию и эксплуатационные расходы. 2) Денежное выражение величины ресурсов, фактически затраченных или потенциально необходимых для достижения определенной цели: производства единицы продукции, выполнения заказа, реализации проекта и т.д. Различаются прямые и косвенные затраты. Прямые затраты непосредственно, а косвенные затраты только каким-либо косвенным путем могут быть отнесены на объект или на центр затрат.

Защита <информации>

Сохранение информации и данных так, чтобы не допущенные к ним лица или системы не могли их читать или изменять, а допущенные лица или системы не ограничивались в доступе к ним.

Защита информационной инфраструктуры

Функция организации, направленная на обеспечение: а) гарантий эффективного и безопасного функционирования систем, с адекватным уровнем конфиденциальности, целостности и доступности, б) информационной безопасности с адекватным уровнем риска и затрат. Защите подлежат такие элементы информационной инфраструктуры, как информация, конкретные системы (главные приложения, обеспечивающие системы, жизненно-важные системы), группы систем для поддержки конкретных операционных программ или сами такие программы (например, контроль авиационного трафика, медицинское страхование и др.).

Инвестирование в ИТ

Намерение организации затратить ресурсы или фактически осуществленные затраты ресурсов на избранные информационные технологии или связанные с ИТ инициативы с ожиданием, что выгода от этих расходов превысит стоимость затраченных ресурсов.

Информационная безопасность

Функция используемых информационных технологий, направленная на обеспечение целостности информационных ресурсов, их доступности, конфиденциальности, подотчетности и достоверности.

Информационная система (ИС)

1) Система, предназначенная для сбора, передачи, обработки, хранения и выдачи информации потребителям и состоящая из следующих основных компонентов: Программное обеспечение; Информационное обеспечение; Технические средства; Обслуживающий персонал. 2) Организованный сбор, обработка и распространение информации, в соответствии с определенными процедурами, автоматизированными или выполняемыми вручную. 3) Комплекс, включающий вычислительное и коммуникационное оборудование, программное обеспечение, лингвистические средства и информационные ресурсы, а также системный персонал и обеспечивающий поддержку динамической информационной модели некоторой части реального мира для удовлетворения информационных потребностей пользователей.

Информационная технология (ИТ)

1) Приемы, способы и методы применения средств вычислительной техники при выполнении функций сбора, хранения, обработки, передачи и использования данных. 2) Компьютеры, вспомогательное оборудование, программное обеспечение, микропрограммы, процедуры, сервисы и связанные с ними ресурсы, используемые организацией для выполнения некоторой функции.

Информационное обеспечение автоматизированной системы

Совокупность системно-ориентированных данных, описывающих принятый в системе словарь базовых описаний (классификаторы, типовые модели, элементы автоматизации, форматы документации и т. д.), и актуализируемых данных о состоянии информационной модели объекта автоматизации (объекта управления, объекта проектирования) на всех этапах его жизненного цикла.

Информационно-коммуникационные технологии (ИКТ)

Совокупность методов, производственных процессов и программно-технических средств, интегрированных с целью сбора, обработки, хранения, распространения, отображения и использования информации в интересах ее пользователей.

ИТ-архитектура

Интегрированная среда для эволюции или поддержки существующих ИТ и приобретения новых ИТ, позволяющая достигнуть стратегических и бизнес-целей предприятия. Полная ИТ-архитектура должна состоять из логической и технической архитектуры. Логическая архитектура обеспечивает высокоуровневое описание миссии предприятия, функциональные и информационные требования, системные компоненты, а также информационные потоки между компонентами. Техническая архитектура определяет конкретные стандарты ИТ и правила, которые будут использоваться для реализации логической архитектуры.

Комплекс средств автоматизации автоматизированной системы (КСА)

Совокупность взаимосогласованных компонентов и комплексов программного, технического и информационного обеспечения, разрабатываемая, изготовляемая и поставляемая как продукция производственно-технического назначения.

Конечный пользователь

Отдельное лицо или группа лиц, которые оперируют системой, когда она пущена в эксплуатацию, в рамках целей, для которых она предназначена.

Конструкция системы

Технический проект системы. На высоком уровне абстракции это может быть некоторая структурная диаграмма. На более подробных уровнях детализации конструкция системы представляется схемами в стиле диаграмм действий, которые

впоследствии будут составлять реализацию логических систем или архитектуру приложений. В объектно-ориентированной нотации это были бы методы и их реализации.

Контрольная точка

1) Важное событие проекта, обычно связанное с достижением важнейших результатов. 2) Ключевое событие проекта, свершение которого является необходимым и достаточным условием, определяющим достижение результатов проекта.

Концептуальная модель данных

Модель данных, используемая для моделирования предметной области информационной системы независимо от технологии, которая будет использоваться для ее реализации.

Логическая модель данных

1) Логическая модель данных определяет представления объектов, о которых предприятие хранит информацию автоматизированным или неавтоматизированным образом. Логическая модель данных могла бы быть представлена как снабженная атрибутами и ключами нормализованная модель сущностей-связей, которая следует из семантической модели предприятия. 2) В современных технологиях баз данных модель данных, определяющая представления полной базы данных или ее части в среде выбранной СУБД с точки зрения пользователя.

Методика

1).Совокупность инструкций (предоставляемых в виде текстов, компьютерных программ, различных инструментов и т.д.), которые обеспечивают пошаговую помощь пользователю. 2) Отраженный документально подход для выполнения действий согласованным, непротиворечивым, учитываемым и повторяемым способом.

Модель

Абстрактное представление реальности в какой-либо форме (в том числе, в математической, физической, символической, графической или дескриптивной), предназначенное для представления определенных аспектов этой реальности и позволяющее отвечать на изучаемые вопросы.

Модель бизнес-процесса

Модель фактической деятельности предприятия, которая осуществляется независимо от системных или конструктивных соображений и от каких-либо организационных ограничений. Эта модель может быть представлена как структурированная методо-ориентированная модель, отражающая не только бизнес-процессы, но и их входы и выходы.

Модель данных

Средство моделирования предметной области, а также организации базы данных на различных уровнях (физическом, логическом) ее представления в среде выбранной СУБД. «Материализуется» в виде совокупности языка описания данных и языка манипулирования данными.

Модель жизненного цикла

Структура, состоящая из процессов, работ и задач, включающих в себя разработку, эксплуатацию и сопровождение программного продукта, охватывающая жизнь системы от установления требований к ней до прекращения ее использования.

Модель информационной системы

В Руководстве по разработке федеральный архитектуры предприятия строка, относящаяся к взгляду проектировщика. Определяет логическую модель данных, архитектуру приложений, а также архитектуру географической дислокации системы. Это понятие соответствует одной из строк матрицы Дж. Захмана.

Модель приложения

Один из компонентов системы конструкторских моделей, которые используются для определения приложений федерального предприятия и его интерфейсов. В существующей архитектуре модели приложений определяют те приложения, которые существуют в данный момент времени для управления данными и поддержки бизнес-функций. В целевой архитектуре модели приложений определяют приложения, которые будут необходимы для управления данными и поддержки будущих бизнес-функций.

Мониторинг

Один из доменов, представляющих группу процессов в классификации верхнего уровня в стандарте CobiT. Этот домен введен в связи с тем, что все процессы ИТ должны регулярно оцениваться с точки зрения их качества и соответствия требованиям контроля.

Надежность информации

Обеспечение менеджмента соответствующей информацией для управления существом дел и выполнения его обязанностей по финансовой и другой отчетности.

Обеспечение качества

Все запланированные и систематически выполняемые в рамках системы качества работы; при необходимости объективные доказательства, обеспечивающие уверенность в том, что объект будет полностью соответствовать установленным требованиям качества.

Ограничение

Условие или лимитирующее правило, налагаемое на систему, которое исходит изнутри либо извне рассматриваемой системы.

Описание архитектуры

Представление архитектуры или проекта, созданное в соответствии с Руководством по разработке федеральной архитектуры предприятия.

Определение данных

Определение всех информационных объектов, специфицированных в физической модели данных. Это определение представлено средствами языка описания данных.

Организационно-методическое обеспечение автоматизированной системы

Совокупность документов, определяющих: организационную структуру объекта и системы автоматизации, необходимых для выполнения конкретных автоматизируемых функций; деятельность в условиях функционирования системы, а также формы представления результатов деятельности.

Оценка

1) Систематическое определение степени соответствия объекта установленным критериям. 2) Экспертиза, проведенная обученной командой профессионалов для определения состояния текущих процессов организации и выявления стоящих перед ней высоко приоритетных вопросов, связанных с процессами, Результатом оценки может стать также организационная поддержка улучшения процессов.

Поведение элемента системы

Характеристика элемента, определяющая каким образом он действует и реагирует.

Предприятие

1) Группа организаций, руководствующаяся общими целями и задачами для предоставления продуктов и услуг. 2) Та часть организации, которая ответственна за приобретение и поставку продуктов и/или услуг, в соответствии с соглашением. Организация может быть вовлечена в несколько предприятий, и предприятие может вовлекать одну или более организаций. 3) Одна или более организаций, совместно выполняющих определенную миссию и руководствующихся общими целями и задачами для предоставления некоторого выхода, например, продукта или услуги.

Прикладная система

Система, обеспечивающая поддержку бизнес-процессов и управленческой деятельности на предприятии.

Принцип

Компонент стратегического направления. В рамках федеральной архитектуры предприятия принципы рассматриваются как утверждения, представляющие стратегическое направление поддержки федерального видения. Кроме того, они являются руководством для принятия решений, служат средством для установления

истины при урегулировании различных разногласий и обеспечивают базу для рассредоточенного и одновременно интегрированного принятия решения. Принципы формализуют правила, ограничения и способы поведения, которые подразделения должны соблюдать в своей повседневной деятельности в течение длительного периода времени, а также устанавливают привилегированное направление действий.

Программное обеспечение автоматизированной системы

Совокупность программ на носителях информации с программной документацией.

Программный продукт

Набор машинных программ, процедур и, возможно, связанных с ними документации и данных.

Проект

1) Уникальный процесс, состоящий из набора взаимоувязанных и контролируемых работ с датами начала и окончания и предпринятый, чтобы достичь цели соответствия конкретным требованиям, включая ограничения по времени, затратам и ресурсам. 2) Попытка с определенными датами начала и окончания, предпринятая для создания продукта или услуги, соответствующих специфицированным ресурсам и требованиям. 3) Целенаправленная деятельность временного характера, предназначенная для создания уникального продукта или услуги. 4) Уникальный комплекс взаимосвязанных мероприятий для достижения заранее поставленных целей при определенных требованиях к срокам, бюджету и характеристикам ожидаемых результатов.

Производительность

Обеспечение информации при оптимальном (наиболее продуктивном и экономичном) использовании ресурсов.

Производственный фактор

Фактор, необходимый для преобразования, транспортировки, хранения и проверки сырья, деталей, узлов (подузлов) и конечных продуктов.

Процесс

1) Набор взаимосвязанных работ, которые преобразуют исходные данные в выходные результаты. 2) Набор взаимосвязанных или взаимодействующих работ, которые преобразуют входы в выходы.

Реверс-инжиниринг

1) Поддержание моделей в течение определенного периода времени с тем, чтобы избежать необходимости «изобретать колесо» при развертывании близких проектов, либо для обновления существующей модели предприятия. 2) Деятельность, направленная на восстановление спецификаций программной или информационной

системы на основе имеющейся ее реализации. Потребность в этом возникает чаще всего в связи с необходимостью модернизации унаследованных систем, для которых полностью или частично утрачена проектная документация, или для осуществления миграции их функциональности в новую среду.

Репозиторий архитектуры

Информационная система, используемая для того, чтобы хранить и обеспечивать доступ к архитектурной информации, а также к информации о взаимоотношениях между информационными элементами и об архитектурных продуктах.

Ресурс

1) Персонал, сырье и материалы, финансы и др., необходимые для получения требуемых результатов. 2) Сущности предприятия, которые обеспечивают некоторые или все возможности, требуемые для функционирования предприятия и/или выполнения бизнес-процессов. 3) Средства, которые потребляются во время исполнения процесса.

Сегмент архитектуры

Представление конкретного предприятия в общей федеральной архитектуре предприятия, имеющее дело, например, с архитектурой общих административных систем или с областями главных программ, такими как торговля или предоставление грантов. Каждый сегмент архитектуры охватывает как существующую, так и целевую архитектуру в рамках сферы действия данного сегмента. Сегмент архитектуры является главной областью бизнеса полного федерального предприятия. Он может рассматриваться как управляемый событиями процесс, который пронизывает федеральное предприятие и должен обладать общностью процесса, данных, цели и приложения для того, чтобы гарантировать рассмотрение вопроса о включении его в федеральную архитектуру предприятия.

Сегментно-архитектурный подход

См. Подход, основанный на сегментной архитектуре

Семантическая модель

Модель реальных объектов, значимых для предприятия. Обычно семантическая модель представляется в форме модели сущностей-связей и определяет выражаемые понятия в виде наиболее значимых целей/стратегий бизнеса, которые позднее реализуются как бизнес-правила.

Сетевая архитектура

Специфическое определение адресов и идентификации линий.

Система

1) Совокупность компонентов, организованных для выполнения определенной функции или набора функций. 2) Комплекс, состоящий из процессов, технических и программных средств, устройств и персонала, обладающий возможностью удовлетворять установленным потребностям или целям. 3) Комбинация взаимодействующих элементов, организованная для достижения одной или более объявленных целей. 4) Совокупность объектов реального мира, организованная для заданной цели.

Согласованность

Соответствие бизнес-процессов законам, правилам и договорным обязательствам, для которых они являются субъектами, т.е. предъявляемым извне критериям бизнеса.

Соответствие <стандарту>

Удовлетворение требований стандарта характеристиками продукта, модели, архитектуры и т.п.

Стандарт

Один из компонентов Руководства по разработке федеральной архитектуры предприятия (FEAF). Стандарты представляют собой совокупности критериев (некоторые из которых могут быть обязательными), рекомендации, подготовленные в инициативном порядке на общественных началах, и материалы, описывающие передовой опыт. Примерами стандартов являются: разработка приложений; управление проектами; управление поставщиками; производственные операции; поддержка пользователя; управление активами; оценка технологии; руководство архитектурой; управление конфигурацией; решение проблем и др.

Структура

1) Структурная диаграмма, которая связывает составные части некоторой концептуальной сущности между собой. 2) Логическая структура для классификации и организации сложной информации.

Существующая архитектура

См. Базовая архитектура

Техническая архитектура

Аспект архитектуры предприятия, характеризующий федеральные данные, используемые приложения и технологии, которые требуются для поддержки потребностей бизнеса предприятия. Существующая техническая архитектура определяет реализованную конструкцию предприятия, которая поддерживает текущие потребности бизнеса. Целевая техническая архитектура определяет возможности, необходимые для поддержки будущих потребностей бизнеса.

Техническое обеспечение автоматизированной системы

Совокупность средств реализации управляющих воздействий, средств получения, ввода, подготовки, преобразования, обработки, хранения, регистрации, вывода, отображения, использования и передачи данных с конструкторской и эксплуатационной документацией.

Технологическая архитектура

Базовая архитектура предприятия. Это модель, являющаяся физическим описанием технологической среды предприятия, показывающим фактическое аппаратное и системное программное обеспечение узлов сети и линий, в том числе операционные системы и программное обеспечение промежуточного уровня.

Технологическая модель

Модель базовых технологий, которая определяет существующую и целевую технологическую архитектуру. Для существующей архитектуры технологические модели определяют существующие в настоящее время технологии, обеспечивающие необходимую среду для работы систем, которые управляют данными и поддерживают бизнес-функции. Для целевой архитектуры технологические модели определяют технологии, которые потребуются для будущей такой среды.

Традиционный архитектурный подход

Подход, при котором в первую очередь разрабатывается регламент для будущего описания архитектуры. Затем должна быть описана существующая техническая база и после этого представлена целевая архитектура. Лишь после того, как все эти действия закончены, начинается проектирование и разработка необходимой архитектуры. Такой подход требует существенных начальных финансовых инвестиций, а также значительного времени для достижения конечной цели.

Унаследованная система

Система, основанная на морально устаревших технологиях, которая уже существовала либо находилась в стадии развертывания на начальном этапе осуществления программы модернизации. На любую унаследованную систему программа модернизации предприятия будет воздействовать в большей или в меньшей степени. Некоторые из них будут использоваться в качестве переходных систем до того момента, когда они будут ликвидированы. Другие унаследованные системы будут ликвидированы по мере того, как их функции перейдут к модернизированным системам. Еще одна группа унаследованных систем может эксплуатироваться до момента, пока они не прекратят свое существование.

Физическая модель данных

1) Представление объектов предприятия, определяемое возможностями технологии, которая выбрана для реализации. При использовании реляционной технологии физическая модель данных является моделью табличной структуры, необходимой для поддержки реляционной логической модели данных. В объектно-ориентированных нотациях используются модели типа иерархий/ассоциаций классов. 2) В современных технологиях баз данных модель данных, определяющая их представления в среде хранения.

Целевая архитектура

Представляет в контексте стратегического направления желаемое будущее состояние предприятия или «что должно быть сформировано». Целевая архитектура состоит из двух частей: а) из целевой бизнес архитектуры, которая определяет будущие потребности бизнеса предприятия, адресованные к новым и зарождающимся технологиям; б) из целевой технической архитектуры, которая определяет будущие конструкции, которые должны использоваться для поддержки будущих потребностей бизнеса.

Целостность информации

Обеспечение точности и полноты информации, а также ее достоверности в соответствии с ценностями и ожиданиями бизнеса.

Цепь результатов

Причинная или логическая связь между деятельностями и результатами, а также последствиями данной политики, программы или инициативы, для реализации которых эти деятельности предназначены.

Электронное Министерство

Метафора, обозначающая информационное взаимодействие органов государственной власти и общества с использованием информационно-коммуникационных технологий.

Электронный обмен данными для систем государственного управления, коммерции и транспорта

Electronic Data Interchange for Administration, Commerce and Trade (EDIFACT)

Принятая в 1996 г. Советом ООН по экономике и социальным проблемам рекомендация по стандартизации электронного обмена информацией.

Элемент конфигурации

Объект внутри конфигурации, который удовлетворяет функции конечного использования и может быть однозначно определен в данной эталонной точке.

Элемент системы

1) Одна из основных частей системы, имеющая характеристики состояния, поведения и идентификации. 2) Один из элементов множества элементов, составляющих систему.

Эргономическое обеспечение автоматизированной системы

Совокупность взаимосвязанных требований, направленных на согласование психологических, психофизиологических, антропометрических, физиологических характеристик и возможностей человека-оператора, технических характеристик КСА, параметров рабочей среды на рабочем месте.

Эталонное тестирование

Структурированный подход, позволяющий представлять лучшие достижения практики в индустрии, компаниях и в Министерстве и адаптировать их к функционированию данной организации.

Эталонный анализ

См. Эталонное тестирование

8.10 Описание базовых схем данных для сквозных пилотных проектов ИС ЭАР

В данном разделе описаны конкретные базовые схемы, используемые в пилотных проектах ИС ЭАР. Эти схемы удовлетворяют всем обязательным требованиям раздела 5.3.1 «Рекомендации по процессу разработки XML-схем» и раздела 5.3.2 «Рекомендации по стилю XML-схем».

8.10.1 Описание организационных справочников в НСИ ИС ЭАР

Информация организационных справочников в НСИ ИС ЭАР представляется следующими базовыми структурами данных (см. UML-диаграмму):

- оргединицы (OrganizationalUnit) – организации (Organization) и их подразделения (Department), такие как «бухгалтерия», «издательский отдел» и пр.
- персоналии (Person) – сотрудники организаций и подразделений
- информация о штате организаций и подразделений – какие сотрудники состоят в каких оргединицах и на каких должностях (ассоциативная сущность Job).

Для указания должностей используется справочник должностей (JobTitle), за основу которого берётся ОКПДТР (ОК 016-94) - Общероссийский классификатор профессий

рабочих, должностей служащих и тарифных разрядов. Должность может быть указана по коду ОКПДТР, либо текстом, если она отсутствует в классификаторе.

Информация о штатном расписании (в каких подразделениях какие должности присутствуют) в базовых структурах данных опускается, поскольку не все участники обладают данной информацией и немногие нуждаются в ней.

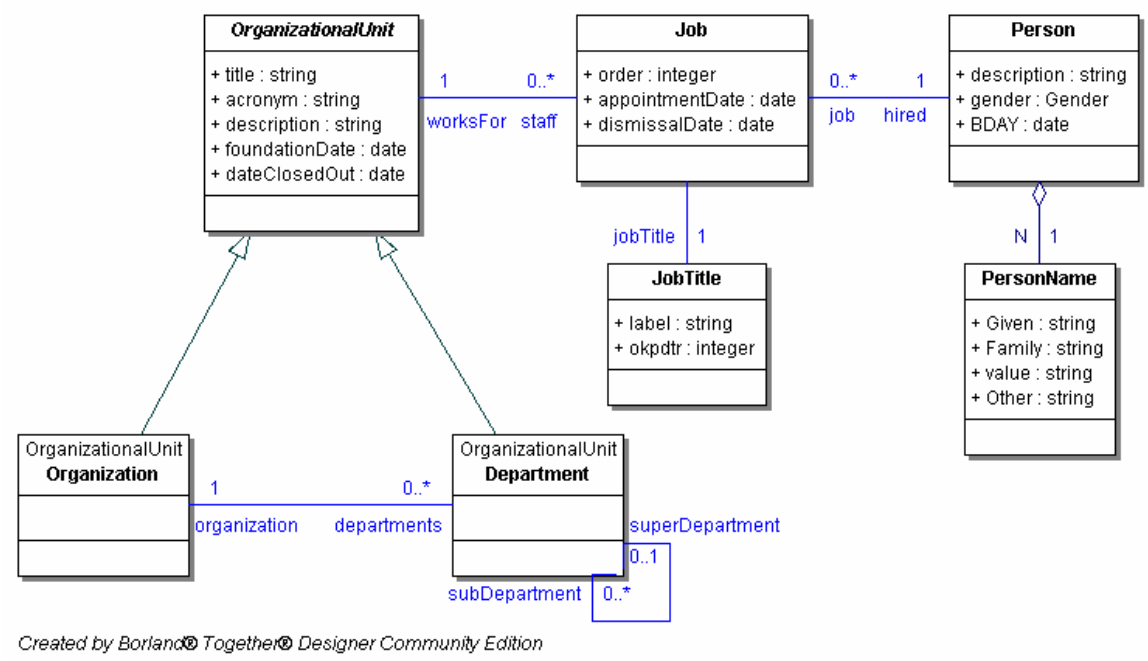


Рисунок 43. Базовые структуры данных ИС ЭАР

8.10.1.1 Состав базовых структур данных

Для приближения описания базовых структур данных к описанию базовых XML-схем данных, при описании состава базовых структур мы будем указывать названия соответствующих им элементов XML-схемы. Ниже перечислены пространства имён элементов схемы:

- xmlns="http://umeta.ru/namespaces/infospace/"
- xmlns:dc="http://purl.org/dc/elements/1.1/"
- xmlns:dcterms="http://purl.org/dc/terms/"
- xmlns:vCard="http://www.w3.org/2001/vcard-rdf/3.0#"
- xmlns:pcv=”http://prismstandard.org/namespaces/1.2/pcv/”
- xmlns:foaf="http://xmlns.com/foaf/0.1/"
- xmlns:ci=" http://umeta.ru/namespaces/blocks/auxilary/contact-information/"

- xmlns:legal="http://umeta.ru/namespaces/infospace/specializations/legal/"
- xmlns:staff="http://umeta.ru/namespaces/infospace/specializations/staff/"

Перечисленные выше базовые структуры данных оргсправочника НСИ ИС ЭАР имеют следующий состав. Почти все свойства не обязательны для указания, обязательными являются только названия, а также связи worksFor/hired в каждой ассоциативной сущности Job.

Информация об организациях (Organization) включает:

- dc:title - наименование
- acronym - сокращённое наименование
- dc:description - произвольное текстовое описание
- foundationDate, dateClosedOut - дату основания и расформирования организации
- staff * - штат данной организации (ассоциативные сущности Job: сотрудник плюс должность)
- department *- подразделения данной организации
- subOrganization *- подчинённые организации
- superOrganization - указание на вышестоящую организацию
- контактная информация:
    - vcard:EMAIL* – email(ы) , строка
    - vcard:TEL* – телефон(ы) , строка
    - vcard:ADR* - адрес(а) , строка
    - ci:otherContact – прочая контактная информация, строка
    - vcard:URL* – Web-страница, ftp и пр., строка
    - ci:contactWith* – контактные персоны, строка
    - ci:howToReach – схема проезда, строка

(звёздочкой отмечены отношения один ко многим и многие ко многим)

Информация о подразделениях (Department) включает:

- dc:title - наименование

- acronym - сокращённое наименование
- departmentCode – код подразделения
- dc:description - произвольное текстовое описание
- foundationDate, dateClosedOut - дату основания и расформирования подразделения
- staff * - штат данной организации (ассоциативные сущности Job: сотрудник плюс должность)
- organization - указание на организацию, к которой относится подразделение
- subDepartment *- подчинённые подразделения
- superDepartment - указание на вышестоящее подразделение, которому подчинено данное подразделение
- контактная информация: как у организаций
- Информация о персоналиях (Person) включает:
- vCard:N - ФИО сотрудника: могут быть отдельно указаны фамилия (vCard:Family), имя (vCard:Given), отчество и пр. (vCard:Other), а также может быть указано ФИО целиком (rdf:value), если оно не может быть разобрано на эти 3 составляющих.
- dc:description - произвольное текстовое описание
- vCard:BDAY – дата рождения
- gender – пол
- job * - места работы (ассоциативные сущности Job: оргединица плюс должность)
- контактная информация человека:

  Замечание: рабочие телефоны и пр. реквизиты должности, а не человека, указываются в описании его связи с организацией (объект «Job»).

  - vcard:EMAIL* – email(ы) , строка
  - vcard:TEL* – телефон(ы) , строка
  - vcard:ADR* - адрес(а) (фактический), строка
  - ci:otherContact – прочая контактная информация, строка
  - foaf:homepage – «домашняя страница» человека, строка

    - foaf:icqChatID – номер ICQ
- birthPlace – место рождения
- официальная информация:
    - legal:inn – ИНН
    - legal:socialInsuranceNumber – номер полиса государственного пенсионного страхования
    - legal:identificationDocument * - подструктуры со следующими полями:
        - legal:documentType – словарь типов, указывается ссылкой по URI
        - legal:documentSeries – серия (строка)
        - legal:documentNumber – номер (строка)
        - dcterms:issued – дата выдачи (дата)
        - dc:publisher – кем выдан (строка)
    - legal:placeOfResidence – адрес регистрации (строка)

Информация о штате организаций (Job) включает:

- hired – указание на сотрудника (Person) данной организации/подразделения
- worksFor – указание на оргединицу, в которой числится сотрудник
- jobTitle – должность сотрудника в данной организации/подразделении.
- appointmentDate – дата назначения сотрудника на данную должность
- dismissalDate – дата увольнения сотрудника с данной должности
- order – порядок сотрудников в списке, отражающий приоритет их должности (начальники раньше, подчинённые - позже)
- контактная информация должности (рабочий телефон, факс и пр.):

  Замечание: домашние телефоны и пр. реквизиты человека, а не должности, указываются в описании человека (объект «Person»).

    - vcard:EMAIL* – email(ы) , строка
    - vcard:TEL* – телефон(ы) , строка

- ci:otherContact – прочая контактная информация, строка
- vcard:URL* – Web-страница, ftp и пр., строка

- информация отдела кадров:
  - staff:personalFileNumber – табельный номер
  - staff:personalFileNumber – номер личного дела
  - staff:monthlySalary – месячная зарплата (в валюте страны, здесь: в рублях)
  - staff:salarySetDate – дата последней смены оклада («последняя дата приказа»)

8.10.1.2 XML-схема оргсправочника НСИ ИС ЭАР

В Приложениях приведена XML-схема оргсправочника НСИ ИС ЭАР, детальная документация к схеме и пример XML-файла с данными, удовлетворяющими этой XML-схеме.

В данном разделе приведём пояснения к XML-схеме оргсправочника НСИ ИС ЭАР.

8.10.1.2.1 Представление структур данных

Структуры данных представляются в виде XSD complex-типов, которым соответствуют одноимённые XML-элементы. XML-документ представляется в следующем формате: корень rdf:RDF, внутри которого перечислены описываемые в документе объекты – организации (элемент Organization), подразделения (элемент Department) и сотрудники (элемент Person). Допускается также вложенное описание объектов (см. «Представление связей между структурами»).

8.10.1.2.2 Представление связей между структурами

При выражении базовых структур данных НСИ ИС ЭАР в XML-виде, возникает необходимость как-то выражать в XML связи между структурами (ассоциации на UML-диаграмме). Принято решение использовать для этого следующие способы:

1. Идентифицировать все свои объекты URI-идентификаторами (RFC 2396) и указывать ссылки на них опять же с помощью URI-идентификаторов. URI используется вместо primary key таблиц и прочих цифровых идентификаторов для исключения конфликта идентификаторов в НСИ ИС ЭАР. URI-идентификатор может быть сформирован на основе такого primary key или иного ключа, но обязательно с префиксом, указывающим «пространство имён» данного ключа, например указывающим организацию-поставщика данных. Для присвоения URI

объекту указывается атрибут *rdf:about*, для ссылки - атрибут *rdf:resource*. Приведём пример использования URI-идентификаторов и ссылок:

```
<Organization rdf:about="http://economy.gov.ru/">
  <dc:title> Минэкономразвития РФ </dc:title>
  <acronym> МЭРиТ </acronym>
  <foundationDate>1967-08-13</foundationDate>
  <dateClosedOut>1967-08-13</dateClosedOut>
</Organization>

<Department rdf:about="http://sample.ru/namespaces/departments/54321">
  <dc:title> Подразделение X </dc:title>
  <foundationDate>1967-08-13</foundationDate>
  <dateClosedOut>1967-08-13</dateClosedOut>
  <organization rdf:resource="http://economy.gov.ru/"/>
</Department>

<Department rdf:about="http://sample.ru/namespaces/departments/54322">
  <dc:title> Подразделение Y подчинённое подразделению X </dc:title>
  <organization rdf:resource="http://economy.gov.ru/"/>
  <!-- ссылка на подразделение X: -->
  <superDepartment
                rdf:resource="http://sample.ru/namespaces/departments/54321"/>
</Department>

<Person rdf:about="http://sample.ru/namespaces/persons/12345">
  <vCard:N>
    <vCard:Given> Иван </vCard:Given>
    <vCard:Other> Иванович </vCard:Other>
    <vCard:Family> Иванов </vCard:Family>
  </vCard:N>
  <vCard:BDAY>1967-08-13</vCard:BDAY>
  <gender rdf:resource="http://umeta.ru/namespaces/infospace/gender/male"/>
  <job>
    <!-- должность - начальник по ОК 016-94 -->
    <jobTitle
                rdf:resource="http://gov.ru/namespaces/classifiers/OK-016-94/21447"/>
    <!-- ссылка на подразделение X: -->
    <worksFor
                rdf:resource=" http://sample.ru/namespaces/departments/54321"/>
  </job>
</Person>
```

2. Вкладывать в XML одно описание внутрь другого, для того чтобы связать их ассоциацией (отношением), например:

```
  <Department>
    <dc:title> Подразделение Z </dc:title>
    <staff>
      <!-- вложенное описание ассоциативной сущности Job в подразделении Z -->
      <jobTitle rdf:resource="http://gov.ru/namespaces/classifiers/OK-016-
94/21447"/>
      <hired>
        <!-- вложенное описание персоналии –
           сотрудника, состоящего в этой должности в подразделении Z -->
        <vCard:N>
          <vCard:Family> Иванов </vCard:Family>
          <vCard:Given> Иван </vCard:Given>
          <vCard:Other> Иванович </vCard:Other>
        </vCard:N>
      </hired>
    </staff>
    <organization rdf:resource="http://economy.gov.ru/"/>
    <subDepartment>
      <!-- вложенное описание подразделения, подчинённого подразделению Z -->
      <dc:title> Подразделение W подчинённое подразделению Z </dc:title>
      ...
    </subDepartment>
  </Department>
```

8.10.1.2.3 Представление двунаправленных связей

Связи между структурами двунаправлены, например, организация – её подразделение, вышестоящее подразделение – подчинённое подразделение, оргединица – сотрудник. В XML достаточно указать связь лишь в одном направлении, связь в обратном направлении подразумевается. Например:

- Если указана связь от организации к подразделению (элемент department в описании Organization), то в описании подразделения указывать элемент organization не нужно.

- Если указана связь от подразделения к организации (элемент organization в описании Department), то в описании организации указывать элемент department со ссылкой на это подразделение не нужно.
- Если указана связь от подразделения к его дочернему подразделению (элемент subDepartment в Department), то в описании дочернего подразделения указывать элемент superDepartment не нужно.
- и пр.

В частности, связь сотрудника с организацией указывается подструктурой, одним из следующих способов (только одним из вариантов):

1. В описание человека вкладывается подструктура «job» – описание его присутствия в должности («место работы»), и указывается ссылка по URI на оргединицу, в которой сотрудник числится:

```
<Person rdf:about="http://sample.ru/namespaces/persons/12345">
  <vCard:N>
    <vCard:Given> Иван </vCard:Given>
    <vCard:Other> Иванович </vCard:Other>
    <vCard:Family> Иванов </vCard:Family>
  </vCard:N>
  <vCard:BDAY>1967-08-13</vCard:BDAY>
  <gender rdf:resource="http://umeta.ru/namespaces/infospace/gender/male"/>
  <job>
    <!-- должность - начальник по ОК 016-94 -->
    <jobTitle
         rdf:resource="http://gov.ru/namespaces/classifiers/OK-016-94/21447"/>
    <!-- ссылка на подразделение X: -->
    <worksFor
         rdf:resource=" http://sample.ru/namespaces/departments/54321"/>
  </job>
</Person>
```

2. В описании подразделения описываются все его сотрудники с помощью двойного вложения подструктур:

```
<Department>
<dc:title> Подразделение W </dc:title>
<staff>
```

```
    <!-- сотрудник 1 -->
    <order>1</order>
    <!-- должность - начальник по ОК 016-94 -->
    <jobTitle rdf:resource=
                                    "http://gov.ru/namespaces/classifiers/OK-016-94/21447"/>
    <hired>
      <!-- описание человека -->
      <vCard:N>
        <vCard:Family> Петров </vCard:Family>
        <vCard:Given> Петр </vCard:Given>
        <vCard:Other> Петрович </vCard:Other>
      </vCard:N>
      <vCard:BDAY>1967-08-13</vCard:BDAY>
    </hired>
  </staff>
  <staff>
    <!-- сотрудник 2 -->
    <order>2</order>
    <!-- должность, отсутствующая в ОК 016-94 -->
    <jobTitle>
     <pcv:label>заместитель директора
                              по административно-хозяйственной части</pcv:label>
    </jobTitle>
    <hired>
      <!-- описание человека -->
      <vCard:N>
        <vCard:Family> Сидоров </vCard:Family>
        <vCard:Given> Сидор </vCard:Given>
        <vCard:Other> Сидорович </vCard:Other>
      </vCard:N>
      <vCard:BDAY>1977-09-14</vCard:BDAY>
    </hired>
  </staff>
  <organization rdf:resource="http://economy.gov.ru/"/>
</Department>
```

8.10.1.2.4 Справочник должностей

Для указания должностей используется справочник должностей, за основу которого берётся ОКПДТР (ОК 016-94) - Общероссийский классификатор профессий рабочих, должностей служащих и тарифных разрядов. Должность может быть указана следующими способами:

1. URI-ссылкой по коду ОКПДТР в формате:

```
http://gov.ru/namespaces/classifiers/OK-016-94/<5-символьный код по классификатору
```

Например, «Директор (начальник) организации» № 21447 - указывается следующим образом:

<jobTitle rdf:resource="http://gov.ru/namespaces/classifiers/OK-016-94/21447"/>

2. текстом плюс кодом ОКПДТР:

```
<jobTitle>
       <pcv:label>начальник</pcv:label>
       <okpdtr>21447</okpdtr>
</jobTitle>
```

3. только текстом, если должность отсутствует в ОКПДТР:

```
<jobTitle>
             <pcv:label>заместитель директора
                 по административно-хозяйственной части</pcv:label>
</jobTitle>
```

8.10.1.2.5 Многоязычность

Желательно указывать язык данных с помощью атрибута xml:lang. Для указания языка всего XML-документа указывается xml:lang при корневом элементе rdf:RDF.

8.11 Контролируемые словари в НСИ ИС ЭАР

8.11.1 Понятие контролируемого словаря

Контролируемый словарь в самом общем понимании – это набор терминов некоторой предметной области и правил их использования для описания информации при индексировании и обработке. Иначе говоря, контролируемые словари предоставляют возможность добавить метаданные к существующей информации и/или связи в ней.

Наиболее простая, но в то же время часто используемая форма контролируемого словаря – это плоский словарь терминов предметной области и их описаний (с, возможно, добавлением неких специальных атрибутов, таких как код и приоритет). Чаще всего плоские словари используются для группировки некоторого набора ключевых терминов и/или наиболее употребимых фраз словарных статей и добавления к ним расшифровок, определений, описаний.

Список терминов:

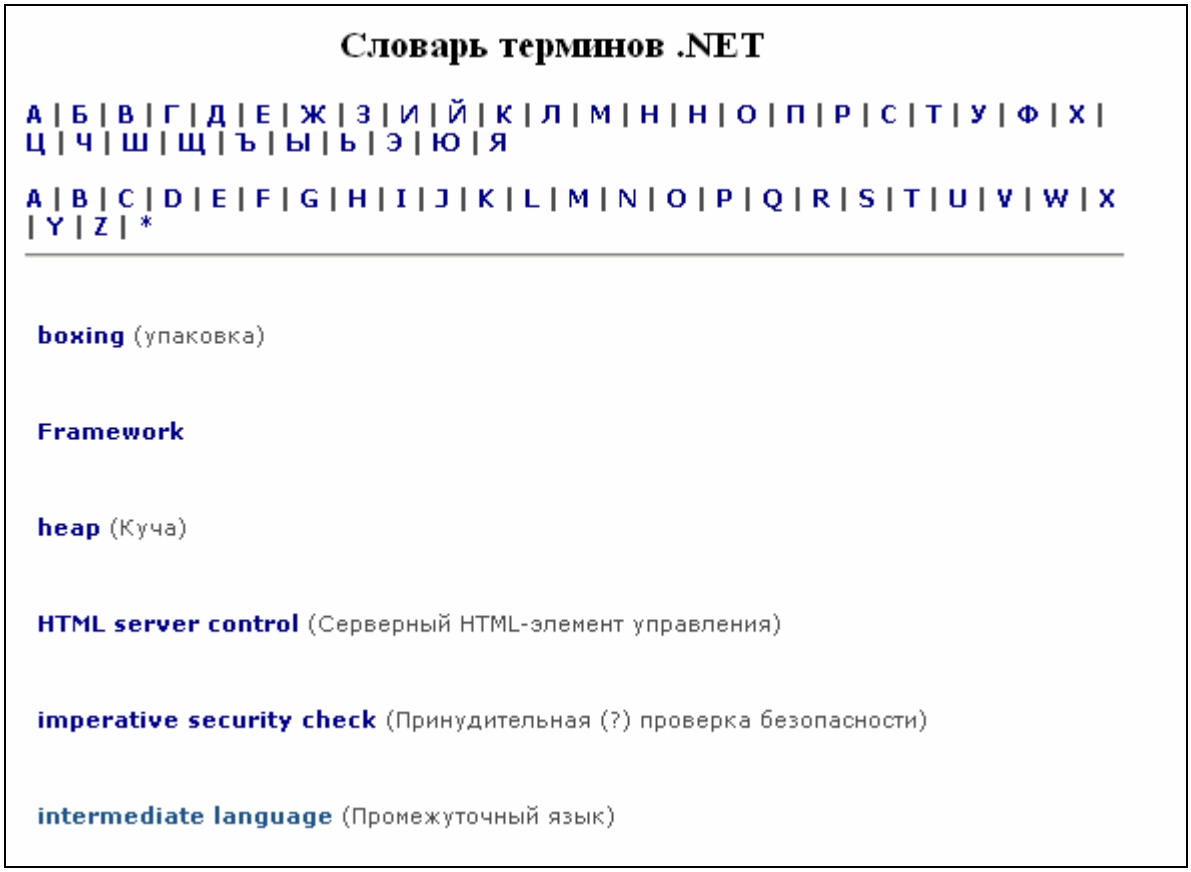


Рисунок 44. Пример списка терминов словаря

Описание термина:

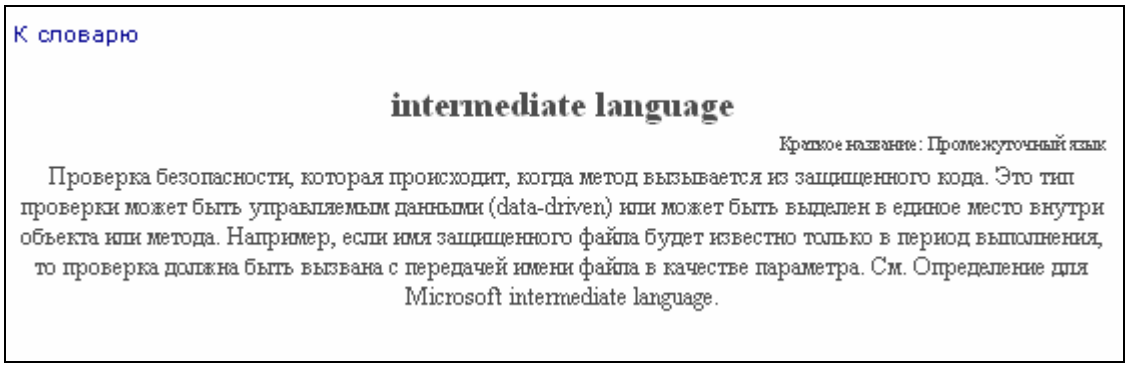


Рисунок 45. Пример информации об элементе словаря

Но плоские словари по своей природе не позволяют отразить связь между понятиями, поэтому для этих целей используется другой вид контролируемых словарей – иерархические классификаторы. Классификатор представляет собой набор терминов с описаниями (и, возможно, другими специальными атрибутами) и связей между ними, чаще всего эти связи древовидную структуру. Как пример такого классификатора можно

привести древовидную структуру географического классификатора ISO3166, описывающего географическое деление стран по регионам.

Рисунок 46. Пример иерархического классификатора

8.11.2 Структуры данных для представления контролируемых словарей в НСИ ИС ЭАР

Информация различных планарных контролируемых словарей в НСИ ИС ЭАР представляется с помощью структур VocabularyTerm, информация иерархических классификаторов – с помощью структур ClassifierTerm.

Ниже представлена UML-диаграмма этих классов и условная OWL-диаграмма.

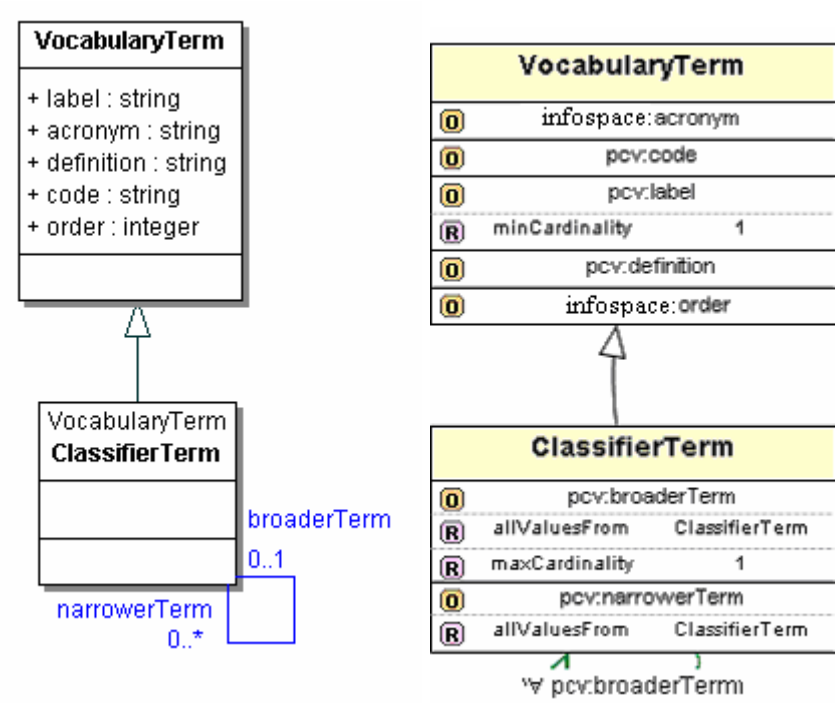


Рисунок 47. UML-диаграмма и OWL-схема, описывающие структуру контролируемых словарей

Каждый конкретный контролируемый словарь (в терминах UML) представляется подклассом VocabularyTerm, конкретный иерархический классификатор – подклассом ClassifierTerm. Элементы (термины) словаря представляются объектами-экземплярами этого класса. В конкретном словаре или классификаторе могут быть введены дополнительные свойства помимо базовых свойств, допустимых во всех словарях.

8.11.3 Состав базовых структур данных для представления контролируемых словарей

Для приближения описания базовых структур данных к описанию базовых XML-схем данных, при описании состава базовых структур мы будем указывать названия соответствующих им элементов XML-схемы. Ниже перечислены пространства имён элементов схемы:

- xmlns:infospace="http://umeta.ru/namespaces/infospace/"
- xmlns:cv="http://umeta.ru/namespaces/blocks/auxilary/controlled-vocabularies/"
- xmlns:pcv=”http://prismstandard.org/namespaces/1.2/pcv/”

При описании каждого элемента контролируемого словаря (cv:VocabularyTerm, cv:ClassifierTerm) допускается указание следующей информации:

- pcv:label – название термина (указывается обязательно)
- infospace:acronym – сокращённое название термина (если применимо)
- pcv:definition – описание термина (если применимо)
- pcv:code – ключ-код, идентифицирующий элемент словаря (если применимо)
- infospace:order – порядок элементов в словаре (если применимо)

Если же словарь является не планарным словарём (cv:ClassifierTerm), а иерархическим классификатором (cv:ClassifierTerm), то при описании каждого термина указываются также его положение в иерархии:

- pcv:broaderTerm – вышестоящий термин в дереве классификатора (более широкое понятие)
- pcv:narrowerTerm * - вложенные термины в дереве классификатора (более узкие понятия) (отношение один ко многим)

В конкретном словаре или классификаторе могут быть введены дополнительные свойства помимо этих базовых свойств.

8.11.4 XML-схема для представления контролируемых словарей НСИ ИС ЭАР

В Приложениях приведена XML-схема для представления контролируемых словарей НСИ ИС ЭАР, детальная документация к схеме и пример иерархического XML-словаря с данными, удовлетворяющими этой XML-схеме (ISO3166 – словарь стран).

См. также раздел «Словари с URI-идентификацией вместо enumerated-типов» в «Руководстве по стилю XML-схем» для обоснования выбора описанной методики представления контролируемых словарей и примера указания значения из словаря.

8.12Адаптеры ИС

8.12.1 Общие требования к правилам формирования адаптеров ИС

Под адаптером АИС в ИСЭАР понимается модуль, обеспечивающий техническую и синтаксическую интероперабельность данной АИС с и ИСЭАР. Адаптер позволяет использовать АИС в деловых процессах ИСЭАР.

Различаются два вида требований к адаптерам ИС:

1. для внешней ИС
2. для ИС локальной сети

Адаптер внешней АИС должен обеспечивать техническую интероперабельность средствами протокола SOAP, то есть внешняя ИС для ИСЭАР должна представлять собой Web-сервис с доступной для ИСЭАР WSDL-спецификацией интерфейса, и синтаксическую интероперабельность средствами XML формата данных, которые должны соответствовать доступной для ИСЭАР XSD-схеме данных.

В этом случае различаются два вида архитектуры адаптера:

1. "тонкий" - внешняя ИС должна только иметь SOAP-клиент, который обращается к Web-сервису, реализуемому ИСЭАР.
2. "толстый" - внешняя ИС должна иметь SOAP-сервер - полностью реализовывать Web-сервис.

Адаптер внешней АИС могут быть реализованы с помощью любой платформы, обеспечивающей поддержку упомянутых технологий.

Адаптер АИС локальной сети может отступать от требований к внешней ИС. Эта гибкость ИСЭАР обеспечивается ее служебными и вспомогательными средствами, реализованными либо в рамках создания ИСЭАР, проектов, реализуемых с его помощью, либо на основе возможностей используемого интеграционного сервера (в рамках работ 2004 года в качестве платформы интеграции используется Microsoft BizTalk Server 2004).

Для технической интероперабельности с ИСЭАР адаптер АИС локальной сети он может использовать файловую систему, протоколы Microsoft Message Queuing (MSMQ), SMTP, POP3, HTTP, Microsoft интерфейсы к РСУБД. Синтаксическая интероперабельность с ИСЭАР может быть обеспечена для ряда структурированных форматов (csv, fix) текстовых файлов на основе настройки соответствующих средств BizTalk Server 2004.

Обязательными требованиями являются только требования к адаптеру внешней АИС. Остальные, опциональные требования, выдвигаются с целью повышения гибкости реализации, снижения издержек при реализации адаптеров и ряда других факторов, которые можно учесть в локальной сети.

ИСЭАР содержит вспомогательные средства поддержки реализации адаптеров, в частности, за счёт средств ЭАР и BizTalk Server 2004, позволяющих организовать вспомогательный рабочий процесс, отвечающий за преобразование форматов и протоколов.

9 Интеграция ИС ЭАР с внешними ИС

Данный раздел состоит из трех частей. В первой части описывается текущая реализация интеграционных механизмов ИС ЭАР.

Вторая часть данного раздела регламентирует набор требований к сторонним информационным системам, выполнение которых при разработке адаптера этой системы позволяет ей полноценно интегрироваться в ИС ЭАР и участвовать в общем процессе межведомственного взаимодействия.

В последней части раздела приводится перечень этапов работ, которые необходимо провести для интеграции ИС в ИС ЭАР. В этом разделе так же указываются требования к квалификации участвующих в процессе интеграции сотрудников и приблизительные трудозатраты.

9.1 Описание существующей реализации механизмов интеграции ИСЭАР с внешними ИС

9.1.1 Основные технические решения по интеграции разнородных информационных систем

Под ИСЭАР понимается комплекс технических и программных средств, обеспечивающих взаимодействие различных систем и приложений благодаря единому узлу интеграции – репозиторию данных и серверу интеграции. Схематически структура среды исполнения электронных административных регламентов показана на рисунке.

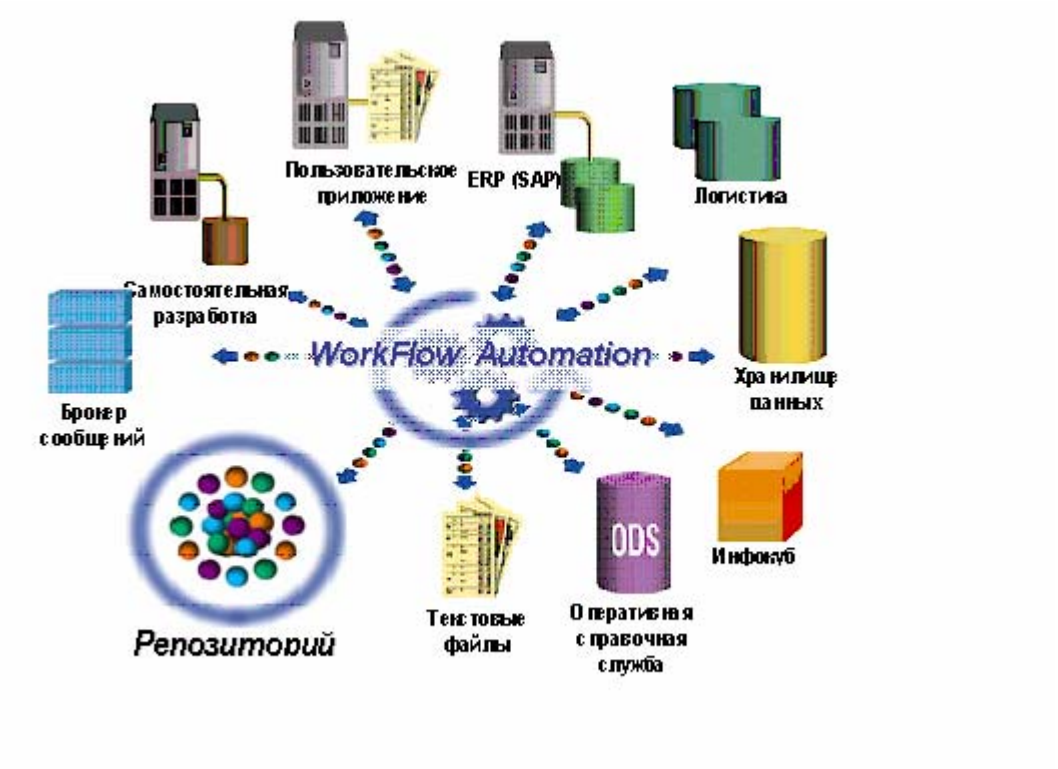


Рисунок 48. Среда исполнения ЭАРов

9.1.1.1 Назначение ИС ЭАР

Исполняющая система электронных административных регламентов (далее ИСЭАР) должна обеспечить механизм исполнения электронных административных регламентов и стать основным связующим звеном для взаимодействия и согласованной работы разрозненных автоматизированных систем Заказчика.

Результатом непосредственного взаимодействия АИС является увеличение зависимостей систем друг от друга, а также неконтролируемый рост числа протоколов и интерфейсов взаимодействия между ними (см. рис. 45).

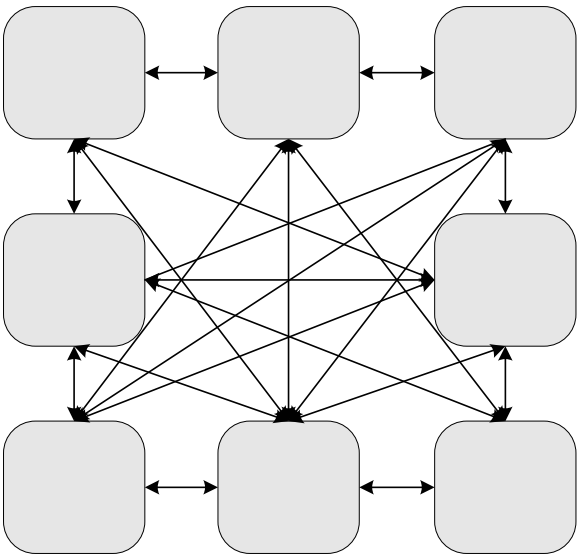


Рисунок 49. Схема взаимодействия автоматизированных информационных систем без участия ИСЭАР

ИСЭАР должна обеспечить унифицированный способ взаимодействия АИС и уменьшить число зависимостей приложений друг от друга (см. рис. 46).

Интеграция является многоуровневым процессом. На уровне базы данных обмениваются информацией разные приложения. На уровне объектного взаимодействия определяется объект, вызываемый, чтобы запустить бизнес-процесс в другой системе. Следующий уровень - это уже взаимодействие посредством бизнес документов и понятий. Поскольку и на уровне данных, и на уровне объектов интеграция весьма затруднительна, то ИСЭАР обеспечивает интеграцию приложений на уровне сообщений (XML-документов).

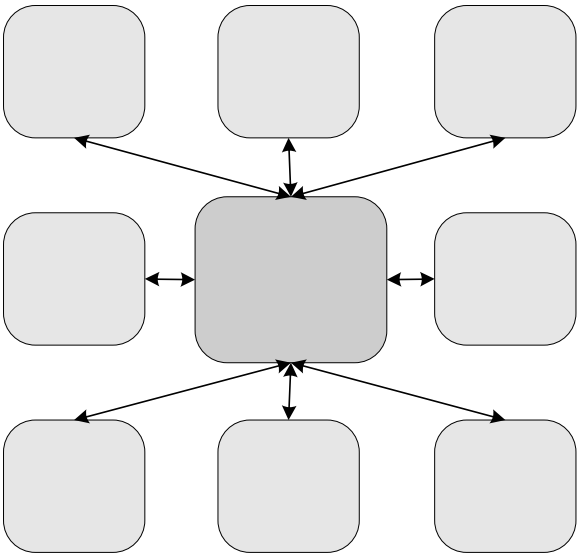


Рисунок 50. Схема взаимодействия автоматизированных информационных систем через ИСЭАР

Создание исполняющей системы электронных административных регламентов должно способствовать достижению как экономических, так и политических целей:

Экономические: снижение издержек бюджета и повышение качества предоставляемых услуг за счет:

- объединения разрозненных источников информации и предоставления унифицированного механизма доступа к этим источникам;
- снижения издержек на интеграцию как уже существующих АИС, так и вновь создаваемых;
- минимизации времени исполнения электронных административных регламентов;
- минимизации возможных злоупотреблений, приводящих, в конечном счете, к росту издержек бюджета.

Политические: обеспечение информационной прозрачности в сфере предоставления госорганами услуг, следствием чего является:

- уменьшение коррупции в этой сфере;
- повышение общественного доверия к органам власти.

9.1.1.2Выбор концепций SOA и Web-сервисов для ИС ЭАР

Функциональные подсистемы представляют собой самостоятельные АИС, предназначенные для автоматизации выполнения части деловых процессов министерства. Важно отметить, что одной из функциональных систем является экономический портал, который обеспечивает взаимодействие с внешними пользователями. В зависимости от типа процессов и требований, выдвигаемых к системе, специфика и параметры разных систем могут существенно отличаться, но для обеспечения большей гибкости и упрощения процесса интеграции, данные системы должны поддерживать SOA (Service Oriented Architecture — «Архитектура, ориентированная на сервисы»).

Архитектура SOA предполагает наличие дискретных программных агентов, которые, взаимодействуя, обеспечивают заданную функциональность. Агенты размещены в разных обрабатывающих средах, поэтому для согласования между ними требуется стек аппаратных и программных протоколов. Очевидно, что это снижает производительность и надежность по сравнению с системами, построенными на прямых вызовах, работающих в

общей аппаратной среде. Кроме того, разработчикам следует учитывать непредсказуемость распределенной среды, в том числе задержки, возможность возникновения конкуренции, вероятный выход из строя части системы. Сущностью, ключевым компонентом SOA являются сообщения, которыми между собой обмениваются агенты и механизм транспортировки, сообщений.

Основные характеристики:

SOA является распределенной архитектурой. Функциональные элементы приложений (модули) могут быть распределены по множеству вычислительных систем и способны к взаимодействию с использованием локальных или глобальных сетей. В частности, Web-сервисы позволяют использовать существующие протоколы, например, HTTP. Модули публикуют свои интерфейсы таким образом, что их использование не требует знания использованных в них решений и технологий; их удобно воспринимать как черный ящик. SOA можно реализовать и без использования Web-сервисов, однако сервисы рассматриваются в качестве основного средства для создания подобной архитектуры.

SOA строится с использованием слабосвязанных интерфейсов. Обычно приложения проектируются в расчете на жесткую связь всех элементов. Как следствие, система должна иметь целостный проект, его изменения в процессе эксплуатации затруднительны. Работу компонентов в слабосвязанных системах проще координировать, системы проще реконфигурировать.

SOA проектируется с ориентацией на процессы (process-centric) с использованием сервисов, каждый из которых ориентирован на решение отдельных задач (task-centric).

SOA содержит два или более уровня. Обращенный к пользователю презентационный уровень отделен от внутреннего, где реализуется бизнес-логика. Последний создается из крупных блоков (coarse-grained chunk), которые обеспечивают сервис клиентским программам презентационного уровня. В логический уровень могут включаться компоненты управления бизнес-процессами (business process management — BPM).

SOA следует реализовывать в виде многозвенной клиент-серверной архитектурой с тонким клиентом.

SOA и Web-сервисы

Как архитектура, ориентированная на сервисы, SOA содержит три обязательных компонента: поставщика сервиса, потребителя сервиса и брокера (см. рисунок 47).

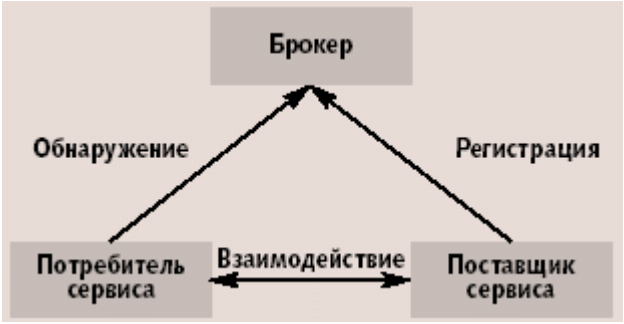


Рисунок 51. Архитектура, ориентированная на сервисы

Поставщик сервиса регистрирует его у некоторого брокера, потребитель там же его обнаруживает, после чего между ними устанавливается связь в соответствии с контрактом и осуществляется требуемое действие. Для создания SOA необходимы три основных типа соглашений:

- транспортное, определяющее форматы и протоколы;
- описание сервиса, для чего нужен некий читаемый машиной язык, на котором описываются выполняемые сервисом операции; после компиляции описания формируется код, определяющий процесс взаимодействия между клиентом и поставщиком сервиса;
- протокол, который определяет способ обнаружения сервиса.

Web-сервис — это реализуемая программными средствами система для поддержки межмашинного взаимодействия через сеть. Интерфейс сервиса описывается на языке, читаемом машиной, например, WSDL. Другие системы взаимодействуют с Web-сервисом способом, указанным в его описании, используя сообщения в стандарте SOAP, передаваемые с использованием HTTP и XML и в сочетании с другими стандартами, относящимися к Web». Физически Web-сервис представляет собой фрагмент программного обеспечения, называемый «агентом». Агент способен передавать и принимать сообщения, он реализует абстрактную функциональность сервиса. Следует проводить различие между агентом и сервисом, один и тот же сервис может быть обеспечен разными агентами.

Web-сервис предназначен для предоставления некоторой функциональности от лица ее владельца (поставщика), это может быть человек или организация (provider entity), которая использует соответствующий собственный агент (provider agent). Запрашивающая сторона (requester entity) тоже может быть человеком или организацией, желающими использовать сервис. Для этого она тоже использует свой агент (requester agent). Агенты взаимодействуют между собой посредством обмена сообщениями. Для

того чтобы этот обмен был успешным, стороны прежде должны достичь соглашения об общем понимании семантики механизме обмена сообщениями.

Механизм обмена сообщениями документирован в описании сервисов (WSD), которое представляет собой читаемую машиной спецификацию интерфейса, включающую: форматы сообщений, типы данных, транспортные протоколы, форматы сериализации, используемые при обмене между агентами заказчика и поставщика услуг. Она также содержит указание на одну или несколько точек в сети (endpoint), откуда поставщик доступен и еще может содержать информацию о предполагаемом шаблоне (pattern) сообщения, используемого для обмена.

Семантика сервиса представляет собой содержание так называемого «контракта» между сторонами; она также содержит некоторые дополнительные сведения о механизмах обмена сообщениями, которые не отражены в WSD. Важно отметить, что, хотя контракт содержит всеобъемлющее представление о сервисе, он не должен быть документально зафиксированным. Контракт может быть явным или скрытым, устным или письменным, в форме, доступной машине или человеку. Различие между описанием сервиса и контрактом заключается в том, что первое определяет механизм взаимодействия с сервисом, а второй — смысл и предназначение этого взаимодействия.

Web-сервисы в основном предназначены для автоматизации процессов управления, которые могли бы быть выполнены вручную, однако в архитектуре Web-сервисов остается место и человеку. Во-первых, его участие требуется при достижении соглашения по семантике и описанию. Человек (или организация) является юридическим владельцем Web-сервисов; именно люди должны согласиться с тем, что данная семантика и данное описание будут управлять взаимодействием между сервисами. Возможно, что со стороны поставщика участие может ограничиться публикацией и предоставлением возможности использовать сервис только в той форме, в какой тот существует (take-it-or-leave-it); иными словами, контракт предполагает неизменяемые условия с его стороны. Есть и другие формы установления отношений, в том числе, через координирующие организации или даже по запросу заказчика. Во-вторых, агенты заказчика и поставщика сервисов создаются людьми, которые отвечают за исполнение соглашений об описании и семантике услуги.

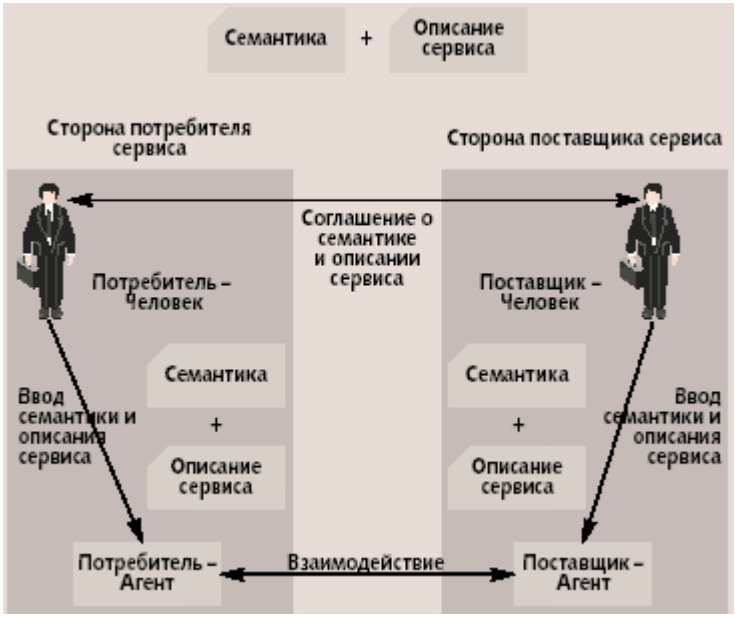


Рисунок 52. Архитектура, ориентированная на сервисы, с учетом предложений WSAWG

9.1.1.3Состав ИС ЭАР

ИСЭАР это, в первую очередь, набор требований к типовому полнофункциональному программному комплексу автоматизации интеграционных, деловых и информационных процессов, включающему средства разработки, спецификации и исполнения этих процессов, необходимые служебные и вспомогательные средства.

С другой стороны, АИС «ИСЭАР» - это сам программно-аппаратный комплекс автоматизации интеграционных, деловых и информационных процессов, для формирования которого могут использоваться как только собственные решения, так решения и средства некоторого стороннего сервера интеграции, его инструментальных и вспомогательных средств и т.п.

В данном разделе рассматриваются требования к общей архитектуре ИСЭАР, приводится перечень подсистем, их назначение и основные характеристики. Состав и функциональное назначение подсистем могут уточняться при доработке и утверждении технического задания, а также на этапе технического проектирования. Сроки разработки и ввода в действие отдельных подсистем определены в разделе 5 настоящего документа и могут уточняться при доработке и утверждении технического задания, а также на стадии технического проекта.

На следующем рисунке приведена структурная схема полнофункциональной АИС «ИСЭАР».

Рисунок 53. Структура ИСЭАР

Разрабатываемая в ходе совокупности этапов полнофункциональная АИС «ИСЭАР» должна включать следующие общеархитектурные подсистемы, которые распределяются по отдельным программным комплексам следующим образом:

- **Инфраструктурный комплекс**.

  Подсистемы комплекса обеспечивают взаимодействие интегрируемых ЭАРом АИС, обмен между ними данными, исполнение ЭАРом

- Подсистема взаимодействия
- Подсистема коммутации сообщений
- Подсистема исполнения ЭАРов
- Подсистема преобразования XML-данных
- Подсистема информационной безопасности
- Подсистема глобальной идентификации ресурсов

- **Комплекс нормативно-справочной информации (НСИ)**.

  Основная цель НСИ - обеспечить формирование и накопление типовой информации, обеспечить циркуляцию в ЭАРах данных, структура и семантика, которых задается типовыми таксономиями и схемами данных. Подсистемы следующим образом группируются.

  - Подсистемы общих справочных данных интегрируемых АИС
    - Подсистема контролируемых словарей и классификаторов
    - Другие подсистемы справочных данных БД НСИ
    - Подсистема репликации и актуализации данных НСИ
  - Подсистемы справочной информации пользователей
    - Подсистема нормативно-правовой информации
    - Подсистема глоссариев
  - Подсистемы метаинформации
    - Реестр ЭАРов
    - Реестр АИС
    - Реестр XML-схем данных и спецификаций онтологий
    - Реестр Web-сервисов

- **Комплекс управления и мониторинга**

  Задача подсистем этого комплекса - поддержка сервисного обслуживания исполнения ЭАРов, регистрация, накопление и анализ информации о прохождении исполнения ЭАРов, и административные функции.

  - Подсистема мониторинга состояния исполняемых ЭАРов
  - Подсистема журнализации действий
  - Подсистема администрирования ИСЭАР
  - Подсистема управления пользователями
  - Подсистема мониторинга состояния системы

- **Комплекс поддержки качества (QA)**

Задача подсистем этого комплекса - поддержка тестирования и проверки качества адаптеров АИС и ЭАРов, и их сертификация по прохождении проверки качества для ввода в эксплуатацию.

- Подсистема отладки и тестирования адаптеров и ЭАРов
- Подсистема верификации и сертификации адаптеров и ЭАРов

- **Комплекс пользовательских интерфейсов**

  Подсистемы комплекса пользовательских интерфейсов обеспечивают взаимодействие пользователей ЭАРов с функциями подсистем ИСЭАР. Основной компонент - “Единое окно” ИСЭАР – представляет собой единый портал для доступа к функциям ЭАРов и ИСЭАР.

  - Подсистема “Единого окна”
  - Справочный интерфейс пользователя

- **Инструментальный комплекс**

  Подсистемы комплекса обеспечивают разработку и описание элементов ЭАРов, поддерживают накопление типовых, шаблонных решений и реализаций адаптеров АИС, web-сервисов и ЭАРов.

  - Подсистемы разработки ЭАРов и BPEL:
    - Подсистема разработки ЭАРов
    - Подсистема BPEL-импорта/экспорта
    - Подсистема моделирования BPEL
  - Подсистема разработки «адаптеров АИС»
  - Подсистема разработки XML-схем данных
  - Подсистема разработки преобразователей XML-данных
  - Подсистема разработки конвейеров

9.1.1.3.1 Инфраструктурный комплекс

Подсистемы комплекса обеспечивают взаимодействие интегрируемых ЭАРом АИС, обмен между ними данными, исполнение ЭАРов.

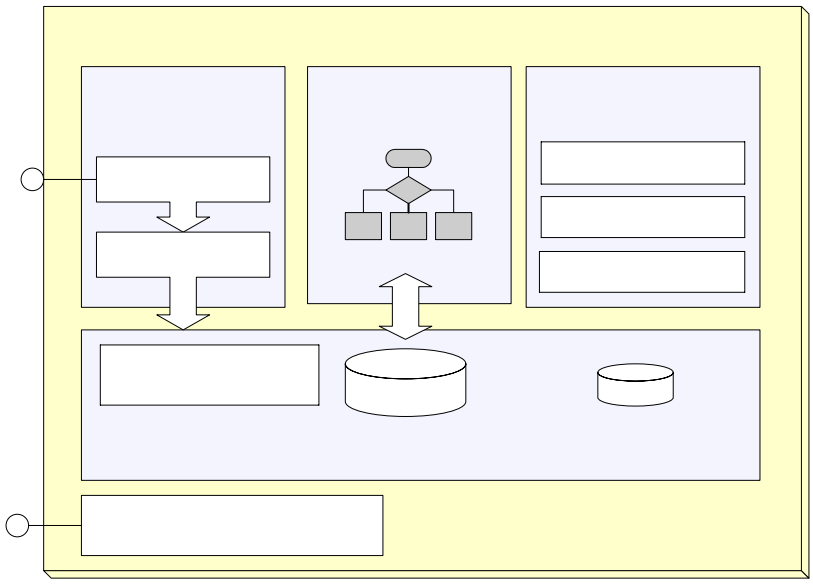


Рисунок 54. Структура инфраструктурного комплекса ИСЭАР

Основой деловых и интеграционных процессов ИСЭАР является обмен между АИС сообщениями. Различные АИС могут использовать различные протоколы и форматы взаимодействия. В одном случае, АИС может предоставлять интерфейс Web-сервиса, то есть предоставлять «внешний адаптер» для взаимодействия с ИСЭАР.

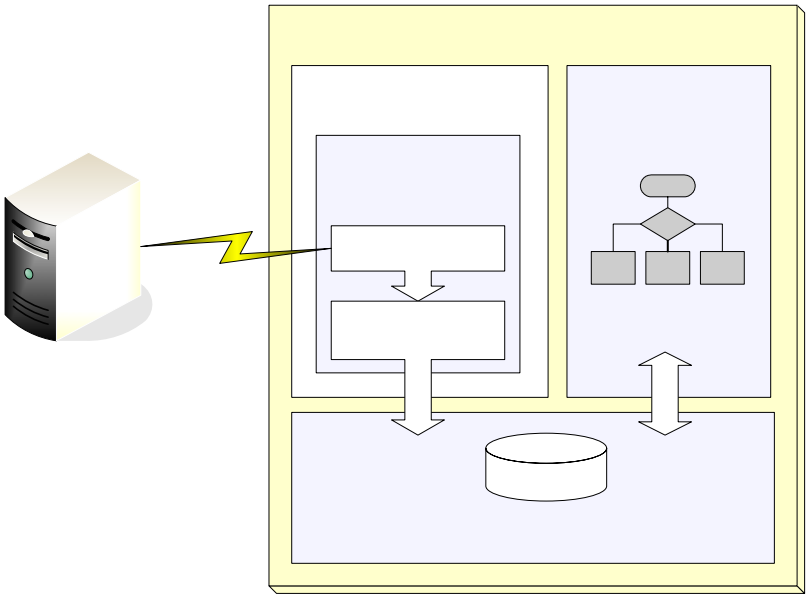


Рисунок 55. Взаимодействие ИСЭАР с АИС локальной сети через «внутренний адаптер»

В другом случае АИС взаимодействует с ИСЭАР с помощью обмена файлами, или почтовыми сообщениями и пр., такой тип взаимодействия с ИСЭАР обозначается условным понятием «внутреннего адаптера» АИС.

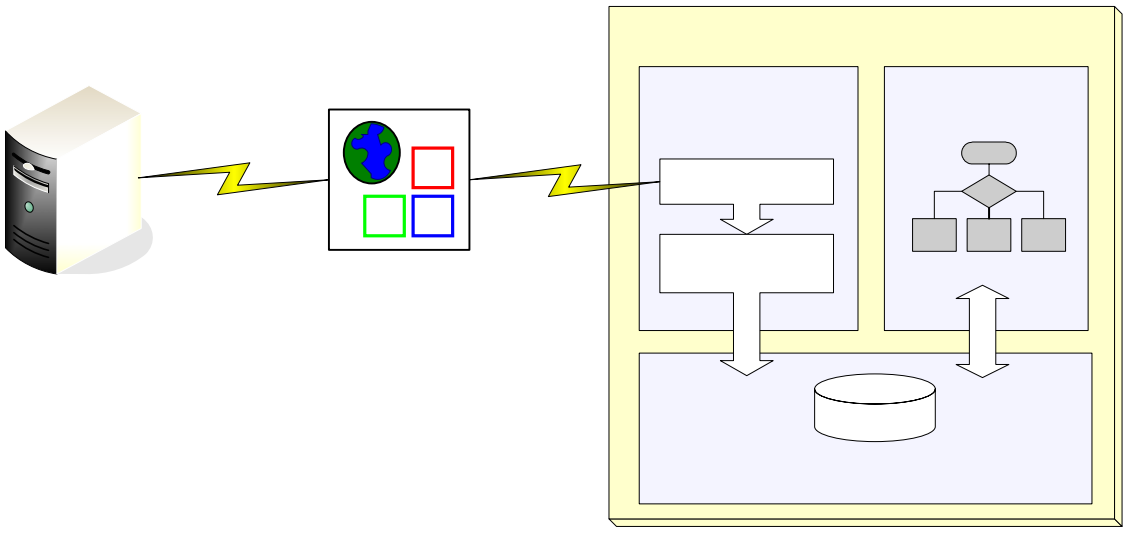


Рисунок 56. Взаимодействие ИСЭАР с внешней АИС через «внешний адаптер»

Всё взаимодействие обеспечивается подсистемой взаимодействия ИСЭАР. Именно эта подсистема отвечает за обмен сообщениями с АИС, поддержку различных протоколов, декодирование и разбор или преобразование данных к единому формату (формату XML) и пр. и передачу сообщений подсистеме коммутации сообщений для вручения их деловым процессам. Аналогично происходит обратный процесс.

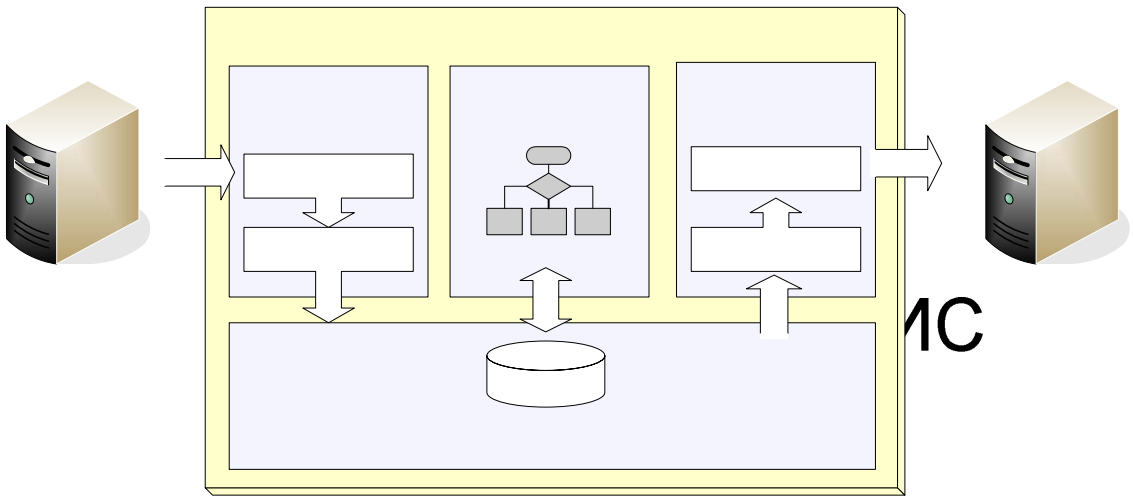


Рисунок 57. Схема функциональной структуры инфраструктурного комплекса ИСЭАР

В целом обработка сообщений в ИСЭАР, как правило, происходит согласно следующей схеме:

1. Сообщение принимается *транспортом* подсистемы взаимодействия. Различные транспорты поддерживают различные механизмы коммуникации, такие как вызов SOAP Web-сервиса, чтение файла, получение почты.
2. Принятое транспортом сообщение проходит обработку в *принимающем конвейере*, которым может содержать компоненты, выполняющие такие функции, как разбор

сообщения из внутреннего формата, предоставленного *отправляющей* АИС, к каноническому формату *XML*-документа, проверка ЭЦП документа и пр.

3. Сообщение передается *подсистеме коммутации сообщений*, которая обеспечивает организацию очереди сообщений и их промежуточное хранение.
4. Согласно спецификации подписок на сообщения, подсистема коммутации сообщений обеспечивает доставку сообщения в целевой деловой процесс, исполняемый *подсистемой исполнения ЭАРов*, который в свою очередь исполняет указанные в нём действия.
5. Результатом действий процесса, как правило, является другое сообщение, оно поступает в *подсистему коммутации сообщений*, она обеспечивает организацию очереди сообщений и их промежуточное хранение и подачу на отправку, согласно спецификациям подписок на сообщения.
6. Сообщение обрабатывается *передающим конвейером*, который может преобразовать его из *XML*-формата, внутренне используемого в ИСЭАР, к формату, необходимому для взаимодействия с *принимающей* АИС. Конвейер может добавить цифровую подпись, закодировать сообщение и пр.
7. Сообщение подаётся на отправку передающему *транспорту*, который использует требуемый протокол/способ взаимодействия для вручения сообщения принимающей АИС.

Помимо применения для обеспечения взаимодействия между ЭАР и АИС в Исполняющей Системе ЭАР, подсистемы инфраструктурного комплекса могут использоваться для построения «внешних адаптеров» АИС на стороне АИС с помощью подсистемы разработки адаптеров АИС.

9.1.1.3.1.1 Подсистема взаимодействия

Модули подсистемы обеспечивают обмен сообщениями между интегрируемыми АИС и ИСЭАР для распространенных протоколов обмена и форматов данных. Подсистема взаимодействия обеспечивает техническую и синтаксическую интероперабельность с различными АИС и служит основой для построения «внутренних адаптеров» АИС на стороне ИСЭАР, а также для построения «внешних адаптеров» АИС на стороне АИС с помощью подсистемы разработки адаптеров АИС инструментального комплекса ИСЭАР.

Подсистема взаимодействия отвечает за обмен сообщениями с АИС, поддержку различных протоколов (техническая интероперабельность), декодирование и разбор или

преобразование данных к единому формату – XML (синтаксическая интероперабельность) и передачу сообщений подсистеме коммутации сообщений.

Функционал подсистемы обеспечивается двумя блоками (наборами) компонент:

- *блоком транспортов*, отвечающих за техническую интероперабельность, поддержку таких протоколов и способов взаимодействия, как SOAP, обмен файлами, обмен электронной почтой, HTTP, FTP и пр.
- *блоком конвейеров*, отвечающих за синтаксическую интероперабельность и конвейерную обработку сообщений, полученных транспортом: декодирование, разбор сообщений и преобразование их к XML-виду, а также дополнительные действия, такие как верификация формата, шифрование-дешифрование, проверка ЭЦП, логгинг и пр.

Конвейеры могут моделироваться графически с помощью средств Подсистемы разработки конвейеров Инструментального комплекса ИСЭАР.

Подсистема взаимодействия содержит встроенный набор транспортов и компонент конвейеров (перечисленный в требованиях к функциям подсистемы), а также обеспечивает возможность расширения этого набора дополнительными компонентами.

В качестве встроенного набора транспортов рассматриваются:

- SOAP-транспорт, обеспечивающий взаимодействие в Web-сервисами с помощью SOAP над HTTP.
- SMTP-транспорт, обеспечивающий перехват отправки почты и пересылку почты.
- POP3-транспорт, обеспечивающий получение почты с почтового сервера (POP3).
- FILE-транспорт, обеспечивающий обмен файлами (чтение/запись файлов из каталогов локальной или сетевой файловой системы).
- HTTP-транспорт, обеспечивающий приём данных на URL сервера ИСЭАР и отправку данных на внешний URL.
- FTP-транспорт, обеспечивающий обмен с FTP-серверами.
- SQL-транспорт, обеспечивающий чтение из БД и запись в БД.

Ниже перечислены допустимые фазы обработки сообщений в принимающих конвейерах и встроенный набор компонент принимающих конвейеров:

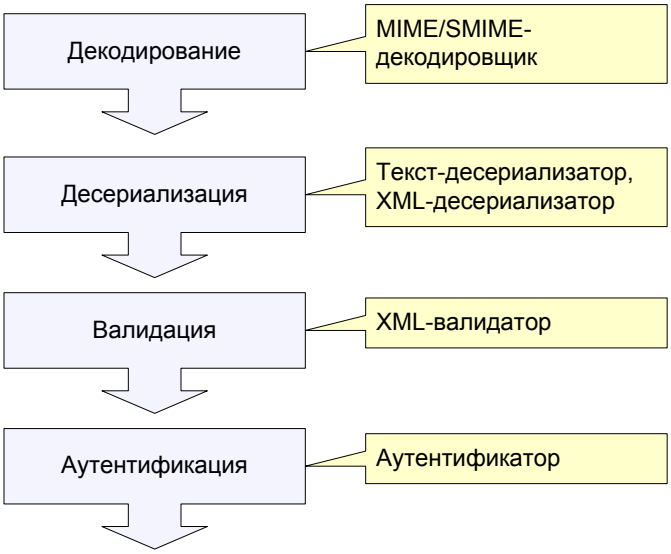


Рисунок 58. Схема принимающего конвейера

- Фаза декодирования:
  - MIME/SMIME-декодировщик, обеспечивающий MIME/SMIME-декодирование, а для SMIME также дешифрование и проверку ЭЦП.
- Фаза десериализации:
  - XML-десериализатор, обеспечивающий разбор документов, представленных в XML-формате.
  - Десериализатор текстовых файлов, обеспечивающий разбор текстовых документов, где данные представлены в виде таблиц с разделителями (например, CSV), либо позиционно (где каждая запись имеет фиксированную длину и структуру), и преобразование этих данных к XML-формату для использования в ЭАРах.
- Фаза валидации:
  - XML-валидатор, обеспечивающий проверку соответствия разобранного XML-файла требуемой XML-схеме
- Фаза аутентификации/определения прав:
  - Аутентификатор (см. подсистему информационной безопасности), обеспечивающий определение пользователя и его прав на основе ЭЦП сообщения, либо на основе идентификатора сессии.

Ниже перечислены допустимые фазы обработки сообщений в передающих конвейерах и встроенный набор компонент передающих конвейеров:

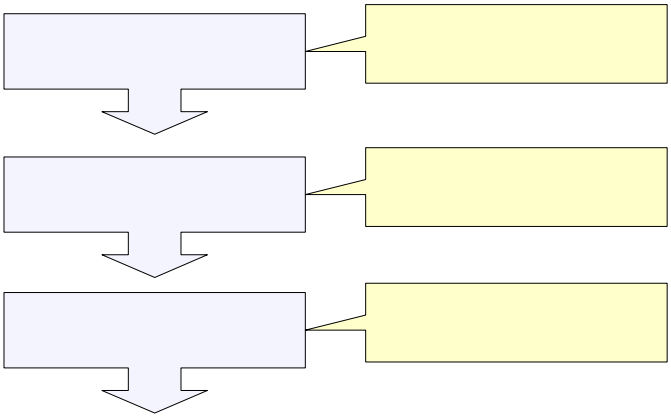


Рисунок 59. Схема передающего конвейера

- Фаза подготовки:
  - Стандартных компонент нет, могут быть запрограммированы и добавлены специфические компоненты.
- Фаза сериализации:
  - XML-сериализатор, обеспечивающий сериализацию в XML-формате, с возможным преобразованием.
  - Сериализатор текстовых файлов, обеспечивающий запись данных, внутренне представленных как XML, в виде текстовых документов, где данные представлены в виде таблиц с разделителями (например, CSV), либо позиционно (где каждая запись имеет фиксированную длину и структуру).
- Фаза кодирования:
  - MIME/SMIME-кодировщик, обеспечивающий MIME/SMIME-кодирование, а для SMIME также шифрование и подпись ЭЦП.

9.1.1.3.1.2 Подсистема коммутации сообщений

Подсистема коммутации сообщений отвечает за приём сообщений от подсистемы взаимодействия, организацию очереди сообщений, их временного хранения и надёжной доставки. Доставка, как правило, производится в деловой процесс (ЭАР), но при использовании инфраструктуры подсистем взаимодействия и коммутации сообщений для построения «внешнего адаптера» АИС, а не для взаимодействия между ЭАР и АИС, сообщения могут сразу подаваться на передачу обратно в подсистему взаимодействия.

Доставка производится согласно правилам коммутации сообщений – подпискам на сообщения. Подписки опираются на такие свойства сообщений, как источник, тип; могут быть указаны условия на содержимое.

Аналогично, при отправке сообщения из ЭАРа, оно передаётся подсистемой исполнения ЭАР в подсистему коммутации сообщений, которая аналогично обеспечивает диспетчеризацию сообщения в подсистему взаимодействия на отправку. Конвейер и транспорт отправки определяются согласно правилам коммутации сообщений – подпискам на сообщения.

9.1.1.3.1.3 Подсистема исполнения ЭАРов

Подсистема исполнения ЭАРов осуществляет исполнение ЭАРа в соответствии с заданной спецификацией ЭАРа. Подсистема исполнения ЭАРов обладает специализированным программным API для взаимодействия с соответствующими подсистемами инфраструктурного комплекса.

Подсистема исполнения ЭАРов обеспечивает поддержку следующих примитивов языка спецификации ЭАРов:

- Приостановку выполнения сценария на определенный интервал времени;
- Приостановку выполнения сценария до наступления определенного момента времени;
- Сигнализация о наступлении исключительной ситуации;
- Обработку исключительных ситуаций;
- Выполнение последовательности действий;
- Выполнение последовательности действий в зависимости от условий;
- Выполнение одного и более действий параллельно;
- Выбор действия при наступлении определенного события.

9.1.1.3.1.4 Подсистема преобразования XML-данных

Подсистема преобразования XML-данных позволяет использовать в ЭАРах преобразования XML-данных от одной XML-схемы в другую, что может использоваться для передачи их другой АИС, их видоизменения или выборочного копирования.

Консорциум W3C разработал язык Extensible Stylesheet Language Transformation (XSLT) как стандартный язык трансформаций XML-документов, и в целях стандартизации и интероперабельности имеет смысл использовать язык XSLT как базис для описания преобразований.

Преобразования могут моделироваться графически с помощью средств Подсистемы разработки преобразователей XML-данных Инструментального комплекса ИСЭАР.

9.1.1.3.1.5 Подсистема информационной безопасности

Подсистема информационной безопасности отвечает за аутентификацию пользователей и авторизацию доступа к защищённым ресурсам, шифрование данных и проверку ЭЦП (с помощью компонент конвейеров подсистемы взаимодействия).

Другой важной чертой информационной безопасности в ИСЭАР является возможность поддержки единой аутентификации (Single Sign-On). Система единой аутентификации включает защищённое хранилище секретной информации, такой как ключи и пароли доступа к различным АИС. Система единой аутентификации сопоставить глобальному идентификатору пользователя его локальные логины и пароли в различных АИС, и отвечает за использование этих паролей для аутентификации в конкретных АИС при взаимодействии с ними через подсистему взаимодействия.

9.1.1.3.1.6 Подсистема глобальной идентификации ресурсов

Все объекты НСИ в должны получать идентификаторы в едином глобальном пространстве имен с гарантией глобальной уникальности каждого идентификатора.

Для облегчения процессов интеграции с внешними АИС подсистема НСИ должна обеспечивать хранение не только собственных идентификаторов НСИ, но и локальных идентификаторов объектов, которые они имеют в АИС, использующих НСИ ИСЭАР.

Подсистема глобальной идентификации ресурсов обеспечивает поддержку глобальной идентификации ресурсов НСИ и интегрируемых АИС, поддержку отображения локальных идентификаторов ресурсов интегрируемых АИС в глобальные идентификаторы и обратно, хранение соответствующих данных.

Средствами подсистемы глобальной идентификации АИС получает возможность:

- запросить ИСЭАР отображение локального идентификатора на глобальный и наоборот
- сообщить НСИ о том, какой локальный идентификатор имеет (получает) глобальный объект при репликации.

9.1.1.3.2 Комплекс нормативно-справочной информации (НСИ)

НСИ представляет собой совокупность реестров разделяемой, семантически богатой, нормативной справочной информации. Разделяемой указывает на то, что эта информация используется широкой совокупностью интегрируемых АИС. Семантически богатой указывает на то, что смысл и назначение информации, известен как использующим ее, так и не использующим ее интегрируемым АИС. Нормативной указывает на то, что при интеграции АИС, их адаптеры обязаны согласовывать свои аналогичные данные с данными, имеющимися в НСИ, в соответствии с назначением и

смыслом последних, что наполнение НСИ должно в большой степени соответствовать имеющимся или складывающимся государственным или корпоративным регламентированным стандартам.

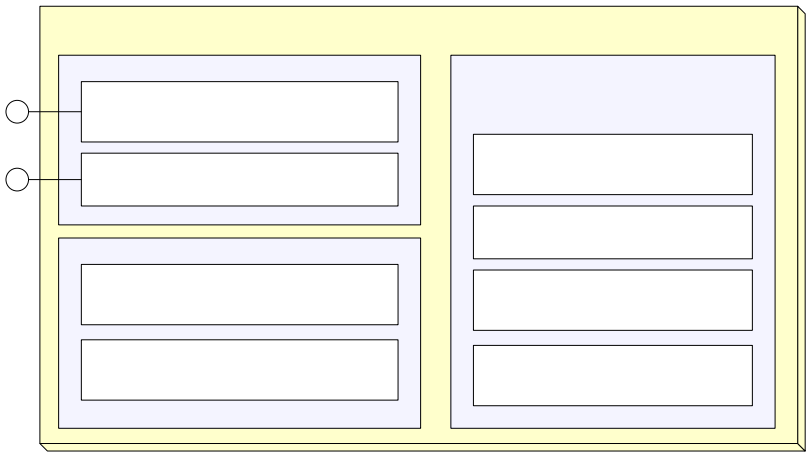


Рисунок 60. Структура комплекса НСИ ИСЭАР

Основная цель НСИ обеспечить формирование и накопление типовой информации, обеспечить циркуляцию в ЭАРах данных, структура и семантика, которых задается типовыми справочниками, схемами данных и онтологиями, предоставить разработчикам ЭАРов эталонные/типовые средства преобразования данных и т.п., чтобы обеспечит снижение издержек при преобразовании данных, реализации и исполнении ЭАР и адаптеров. В соответствии с этим подсистемы следующим образом группируются.

Кроме этого, в комплекс НСИ входят подсистемы справочной информации пользователей, отвечающие за хранение нормативно-правовой документации используемых ЭАРов и пр., и подсистемы метаинформации, обеспечивающие автоматизацию функций динамического расширения ИСЭАР новыми ЭАРами и АИС.

9.1.1.3.2.1 Подсистемы общих справочных данных интегрируемых АИС

Задачами подсистем комплекса являются хранение и управление справочными данными прикладных областей, относящимися к поддерживаемым АИС «ИСЭАР» электронным административным регламентам, и совместно используемых участвующими в ЭАРах АИС. Подсистемы этой группы обозначаются также «подсистемами справочных данных БД НСИ».

9.1.1.3.2.1.1 Подсистема контролируемых словарей и классификаторов

Подсистема контролируемых словарей и классификаторов (по-другому, таксономий) предназначена для хранения, контроля целостности используемых при организации взаимодействия контролируемых словарей и классификаторов.

9.1.1.3.2.1.2 Другие подсистемы справочных данных БД НСИ

Системы базы данных НСИ обеспечивают хранение и управление нормативно-справочными данными, относящимися к поддерживаемым АИС «ИСЭАР» электронным административным регламентам, например, организаций, подразделений министерства, сотрудников подразделений, описаний документов, проектов и т.п..

9.1.1.3.2.1.3 Подсистема репликации и актуализации данных НСИ

Подсистема репликации и актуализации данных НСИ обеспечивает обмен на базе протокола SOAP поступлениями данных между подсистемами справочных данных БД НСИ и АИС, поставляющими данные в НСИ или потребляющих их. Подсистема отвечает разным типам взаимодействия с АИС - «рассылку» данных при наступлении заданного события, «вытягивание» данных при наступлении заданного события, обмен на основе согласования действий уведомлениями.

9.1.1.3.2.2 Подсистемы справочной информации пользователей

В эту группу входят подсистемы справочной информации пользователей, отвечающие за хранение нормативно-правовой документации используемых ЭАРов и пр.

9.1.1.3.2.2.1 Подсистема нормативно-правовой информации

Подсистема нормативно-правовой информации представляет собой хранилище неструктурированных и слабоструктурированных данных справочного, нормативного и правового характера, в частности, для нужд «справочного интерфейса пользователя». Состав информационного наполнения включает:

- хранение нормативно-правовых документов, фигурирующих на различных этапах процесса выполнения ЭАР-ов и их создания;
- хранение нормативно-правовых материалов, описывающих государственные и ведомственные электронные административные регламенты как для открытого доступа к ним потребителей ЭАР-ов, так и для разработчиков ЭАР-ов и экспертов.

9.1.1.3.2.2.2 Подсистема глоссариев

Подсистема глоссариев предоставляет возможность организации глоссариев справочного и нормативно-правового характера, в частности, для нужд «справочного интерфейса пользователя».

9.1.1.3.2.3 Подсистемы метаинформации

Подсистемы этой группы представляют собой набор реестров метаинформации, обеспечивающих автоматизацию и упрощение функций динамического расширения ИСЭАР новыми ЭАРами и АИС.

9.1.1.3.2.3.1 Реестр ЭАРов

Основными функциями реестра ЭАРов является регистрация и изменение описаний ЭАРов, используемых ИСЭАР. Реестр обеспечивает обработку необходимого набора атрибутов для описания ЭАРов.

9.1.1.3.2.3.2 Реестр АИС

Реестр АИС обеспечивает хранение и обработку данных (их описаний и настроек) об АИС, участвующих во взаимодействии с АИС «ИСЭАР» в соответствующих электронных административных регламентах. Реестр обеспечивает обработку необходимого набора атрибутов для описания АИС.

9.1.1.3.2.3.3 Реестр XML-схем данных и спецификаций онтологий

Все структурированные данные, сообщения и документы, обрабатываемые ИСЭАР в ходе реализации интеграционных, деловых и информационных процессов должны представляться в форме XML-документов. Для обеспечения согласованности, целостности передаваемых и обрабатываемых данных все сообщения и документы должны соответствовать XML-схемам в формате W3C XML Schema. Реестр XML-схем данных и спецификаций онтологий обеспечивает хранение и обработку сведений об XML-схемах данных, сообщений и документов, сами XML-схемы, формируемые и обрабатываемые в ходе электронного взаимодействия АИС с АИС «ИСЭАР». Реестр обеспечивает обработку необходимого набора атрибутов для описания схем документов.

9.1.1.3.2.3.4 Реестр Web-сервисов

Основными функциями реестра Web-сервисов является регистрация, изменение и обнаружение описаний Web-сервисов, технических спецификаций их интерфейсов (WSDL). Реестр обеспечивает обработку необходимого набора атрибутов для описаний Web-сервисов.

9.1.1.3.3 Комплекс управления и мониторинга

Задача подсистем этого комплекса - поддержка сервисного обслуживания исполнения ЭАРов, регистрация, накопление и анализ информации о прохождении исполнения ЭАРов, и административные функции.

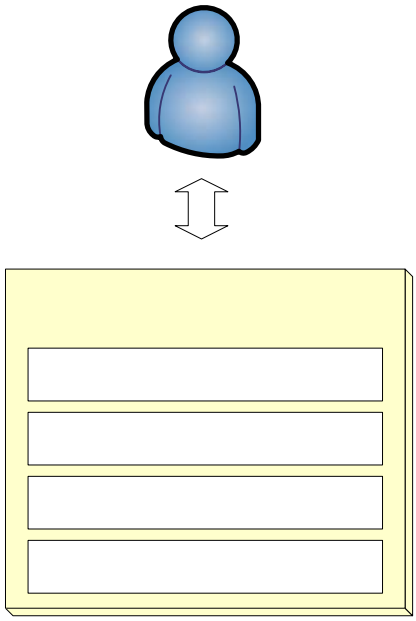


Рисунок 61. Структура комплекса управления и мониторинга ИСЭАР

9.1.1.3.3.1 Подсистема мониторинга состояния исполняемых ЭАРов

Подсистема отслеживает состояния процесса обработки информации, что является весьма существенно, особенно при возникновении ошибок во время исполнении ЭАРов.

9.1.1.3.3.2 Подсистема журнализации действий

АИС «ИСЭАР» в ходе исполнения электронных административных регламентов осуществляет достаточно большое количество разнообразных действий, как то:

- прием и отправка, обработка сообщений АИС «ИСЭАР» с внешними в процессе исполнения ЭАРов;
- запуск и останов сценариев ЭАРов;
- действия пользователей в «Едином окне» АИС «ИСЭАР»;
- получение из внешних АИС данных НСИ, их модификации другими путями;
- предоставление внешним АИС данных НСИ.

Подсистема журнализации действий отслеживает эти события и обсепечивает их журнализацию, в частности, осуществляет регистрацию сообщений, которыми обмениваются интегрируемые АИС, поддерживает архив истории исполнения ЭАРов.

9.1.1.3.3.3 Подсистема администрирования ИСЭАР

Подсистема администрирования служит для управления и конфигурирования подсистем АИС «ИСЭАР» и выполняемых ЭАРов. Подсистема включает три административных инструмента, отвечающих различным нуждам управления и конфигурирования:

- Административную консоль ИСЭАР
- Менеджер конфигурации ЭАРов
- Менеджер подсистемы коммутации сообщений

Консоль администратора ИСЭАР отвечает за управление серверами ИСЭАР и управление ЭАРами (запуск и остановка).

Менеджер конфигурации ЭАРов отвечает за возможность настройки портов приема и отправки сообщений подсистемы взаимодействия, таких как адреса для соединения с Web-сервисами или папки обмена файлами с АИС локальной сети, и прочие параметры ЭАРа, без необходимости изменения сценария делового процесса, использующего данные порты.

Менеджер подсистемы коммутации сообщений обеспечивает обзор и управление подписками подсистемы коммутации сообщений, управляющими коммутацией сообщений.

9.1.1.3.3.4 Подсистема управления пользователями

Подсистема управления пользователями служит для управления пользователями Подсистемы информационной безопасности ИСЭАР. Подсистема отвечает также за хранение дополнительной информации о пользователях, такой как контактная информация, и за отображение пользователей Подсистемы информационной безопасности ИСЭАР на сотрудников организационного справочника БД НСИ.

9.1.1.3.3.5 Подсистема мониторинга состояния системы

Подсистема мониторинга состояния системы отвечает за предоставление администратору ИСЭАР возможности отслеживания общего состояния всего, что происходит в системе, мониторинга исполняемых процессов, возможности выявления ошибок и оценки производительности.

9.1.1.3.4 Комплекс поддержки качества (QA)

Задача подсистем этого комплекса - поддержка тестирования и проверки качества адаптеров АИС и ЭАРов, и их сертификация по прохождении проверки качества для ввода в эксплуатацию.

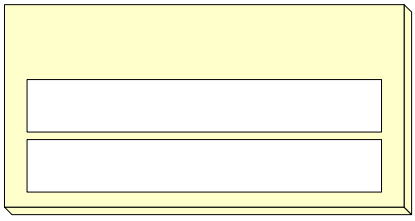


Рисунок 62. Структура комплекса поддержки качества ИСЭАР

9.1.1.3.4.1 Подсистема отладки и тестирования адаптеров и ЭАРов

Подсистема отвечает за поддержку:

- средств отладки сценариев ЭАРов, обеспечивающих управляемую и пошаговую трассировку выполнения сценариев с возможностью интерактивного выполнения соответствующих операций;
- средств тестирования «внешних адаптеров» АИС и Web-сервисов.

9.1.1.3.4.2 Подсистема верификации и сертификации адаптеров и ЭАРов

Подсистема отвечает за поддержку средств верификации и сертификации ЭАРов и адаптеров, подключаемых к АИС «ИСЭАР», интегрируемых в рамках интеграционных и деловых процессов АИС «ИСЭАР».

9.1.1.3.5 Комплекс пользовательских интерфейсов

Подсистемы комплекса пользовательских интерфейсов обеспечивают взаимодействие пользователей ЭАРов с функциями подсистем ИСЭАР. Основной компонент - “Единое окно” ИСЭАР – представляет собой единый портал для доступа к функциям ЭАРов и ИСЭАР.

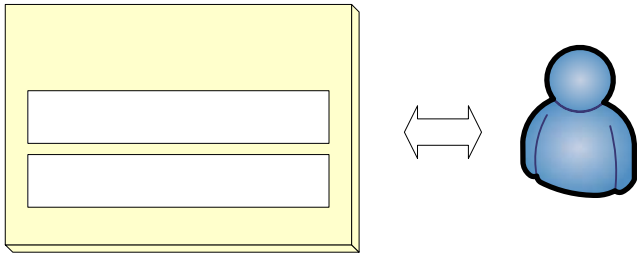


Рисунок 63. Структура комплекса пользовательских интерфейсов ИСЭАР

9.1.1.3.5.1 Подсистема “Единого окна”

Подсистема “Единого окна” обеспечивает прикладной пользовательский интерфейс инициализации, остановки и отслеживания состояния исполнения ЭАРов, поддерживаемых АИС «ИСЭАР». Подсистема “Единого окна” взаимодействует с основными подсистемами инфраструктурного и других комплексов и представляет собой пользовательский портал к сервисам ИСЭАР и её ЭАРов. “Единое окно” поддерживает навигационно-поисковый интерфейс пользователя, позволяющий пользователю найти требуемый ЭАР среди поддерживаемых АИС «ИСЭАР».

9.1.1.3.5.2 Справочный интерфейс пользователя

Справочный интерфейс пользователя обеспечивает информирование пользователей о наличии справочной и нормативно-правовой информации, относящейся к ЭАРам, поддерживаемым АИС «ИСЭАР» или разрабатываемых для нее.

9.1.1.3.6 Инструментальный комплекс

Подсистемы инструментального комплекса обеспечивают разработку и описание элементов ЭАРов, поддерживают накопление типовых, шаблонных решений и реализаций адаптеров АИС, web-сервисов и ЭАРов.

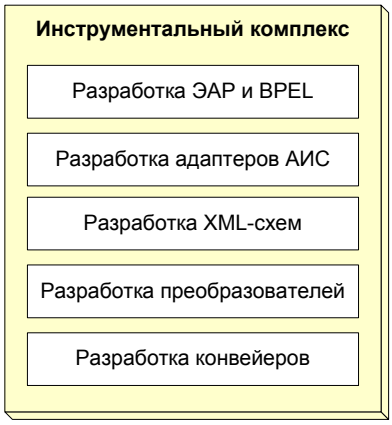


Рисунок 64. Структура инструментального комплекса ИСЭАР

9.1.1.3.6.1 Подсистемы разработки ЭАРов и BPEL

Подсистемы обеспечивают разработку Электронных Административных Регламентов (ЭАРов) и интероперабельность ИСЭАР с другими средами исполнения деловых и интеграционных процессов.

Электронный Административный Регламент описывает деловой или интеграционный процесс, автоматизирующий некоторую деятельность ведомственных Автоматизированных Информационных Систем (АИС). ЭАР взаимодействует с различными АИС через адаптеры этих АИС. Адаптер АИС может быть как «внешним адаптером», то есть следовать архитектуре WSA (Web Services Architecture), так и внутренним адаптером «АИС локальной сети».

Язык описания рабочих процессов Business Process Execution Language for Web Services (BPEL4WS 1.1, или просто BPEL) стал основным стандартом описания композиций Web-сервисов и деловых процессов, в которых участвуют Web-сервисы.

Ввиду гибкости понятия адаптера в ИСЭАР, не всякий адаптер обязан следовать WSA (во многих случаях это в принципе невозможно), и соответственно не всякий ЭАР может быть описан с помощью BPEL.

Подсистема разработки ЭАРов обеспечивает разработку ЭАРов, использующих все функциональные возможности ИСЭАР.

Тем не менее, многие ЭАРы вполне укладываются в функциональные возможности языка BPEL, и в рамках работ по стандартизации имеет смысл обеспечить интероперабельность ИСЭАР с другими средами исполнения деловых и интеграционных процессов с помощью обмена описаниями процессов на языке BPEL. Подсистема BPEL-импорта/экспорта обеспечивает эту интероперабельность.

Следующим шагов в направлении работ по стандартизации и интероперабельности является полная поддержка визуального моделирования WSA-совместимых ЭАРов с помощью примитивов языка BPEL4WS.

9.1.1.3.6.1.1 Подсистема разработки ЭАРов

Целью подсистемы разработки ЭАРов является поддержка разработку ЭАРов, использующих все функциональные возможности ИСЭАР, удобное визуальное моделирование ЭАРов. Существенна также возможность публикации ЭАРа как Web-сервиса, что позволяет использовать ЭАР из внешних по отношению к ИСЭАР систем, в частности, из межведомственной Процедуры Электронного Взаимодействия (ПЭВ), исполняемой Средой Электронного Взаимодействия (СЭВ), а также из внутренних систем и других ЭАРов (компоновка ЭАРов).

9.1.1.3.6.1.2 Подсистема BPEL-импорта/экспорта

Целью подсистемы BPEL-импорта/экспорта является поддержка интероперабельности ИСЭАР с другими средами исполнения деловых и интеграционных процессов с помощью обмена описаниями процессов на языке BPEL4WS 1.1.

9.1.1.3.6.1.3 Подсистема моделирования BPEL

Целью подсистемы моделирования BPEL является поддержка визуального моделирования WSA-совместимых ЭАРов с помощью примитивов языка BPEL4WS 1.1, выгрузка и загрузка этих описаний в формате BPEL4WS 1.1.

9.1.1.3.6.2 Подсистема разработки «адаптеров АИС»

Подсистема упрощает создание адаптеров интегрируемых АИС. Под адаптером АИС в ИСЭАР понимается модуль, обеспечивающий техническую и синтаксической интероперабельность данной АИС с АИС «ИСЭАР». Адаптер позволяет использовать АИС в деловых процессах ИСЭАР.

Например, разработка «внешнего адаптера» включает настройку следующих элементов адаптера:

- предоставление адаптером интерфейса Web-сервиса с помощью SOAP-транспорта Подсистемы взаимодействия;
- взаимодействие с АИС в удобном ей формате с помощью соответствующего транспорта Подсистемы взаимодействия, в частности, обмена файлами;
- преобразование данных внутри адаптера;
- преобразование локальных идентификаторов ресурсов АИС в глобальные идентификаторы с помощью средств Подсистемы глобальной идентификации.

9.1.1.3.6.3 Подсистема разработки XML-схем данных

XML является основным обменным форматом ИСЭАР, а для «внешних» адаптеров – и единственным обменным форматом (через SOAP). Язык XML Schema используется для описания схем XML-документов, схем сообщений, передаваемых между ИСЭАР и различными АИС.

Целью подсистемы разработки XML-схем данных является поддержка визуального моделирования XML-схем.

9.1.1.3.6.4 Подсистема разработки преобразователей XML-данных

Адаптеры и ЭАРы могут осуществлять преобразования XML-данных с помощью Подсистемы преобразования XML-данных Инфраструктурного комплекса.

Целью подсистемы разработки преобразователей XML-данных является поддержка визуального моделирования преобразователей XML-данных, указывающих отображение исходной XML-схемы в целевую.

9.1.1.3.6.5 Подсистема разработки конвейеров

Адаптеры и ЭАРы используют конвейеры Подсистемы взаимодействия для спецификации последовательности действий декодирования, разбора сообщения, полученного из транспорта, к XML-форме для обеспечения синтаксической интероперабельности, а также указания дополнительных действий, таких как верификация формата, шифрование-дешифрование, проверка ЭЦП, логгинг и пр.

Целью подсистемы разработки конвейеров является поддержка визуального моделирования конвейеров Подсистемы взаимодействия.

9.1.2Характерные особенности реализации механизма Web-сервисов для интеграции компонент ИС ЭАР

В рамках работ 2004 года в качестве базиса для ИСЭАР используется сервер интеграции Microsoft BizTalk Server 2004. SOAP-адаптер BizTalk Server позволяет интегрировать в ИСЭАР системы, удовлетворяющие требованиям поддержки

архитектуры WSA и прочим требованиям, рассмотренным в разделах 4. «Вопросы интеграции в контексте Web-технологий» и 5. «Общие рекомендации по использованию стандартов в рамках СУВ».

Для интеграции ИСЭАР с внешними АИС реализуется «адаптер АИС», обеспечивающий интероперабельность АИС в ИСЭАР. Подробнее про адаптеры в ИСЭАР см. раздел «7.12.1 Общие требования к правилам формирования адаптеров ИС». Общая структура интегрируемых с помощью технологии Web-сервисов компонент ИС ЭАР с АИС рассмотрена в разделе 9.1.3.

9.1.3Общая структура интегрируемых с помощью технологии Web-сервисов компонентов ИС ЭАР с АИС

Схема функциональной структуры АИС ИСЭАР показана на Рисунке 50. Эта диаграмма показывает подсистемы ИСЭАР, участвующие во взаимодействии с АИС.

Для интеграции ИСЭАР с внешними АИС реализуется «адаптер АИС», обеспечивающий интероперабельность АИС в ИСЭАР. Подробнее про адаптеры в ИСЭАР см. раздел «7.12.1 Общие требования к правилам формирования адаптеров ИС».

Требования к адаптеру АИС различаются для адаптера внешней АИС и адаптера АИС локальной сети. Адаптер внешней АИС должен удовлетворять требованиям поддержки архитектуры WSA и прочим требованиям, рассмотренным в разделах 4. «Вопросы интеграции в контексте Web-технологий» и 5. «Общие рекомендации по использованию стандартов в рамках СУВ».

Адаптер АИС локальной ведомственной сети может отступать от требований поддержки WSA и упомянутых требований стандартов СУВ. Эта гибкость ИСЭАР обеспечивается ее служебными и вспомогательными средствами, реализованными либо в рамках создания ИСЭАР, проектов, реализуемых с его помощью, либо на основе возможностей используемого интеграционного сервера (Microsoft BizTalk Server 2004). Облегчённые требования к адаптерам локальной сети введены с целью повышения гибкости реализации, снижения издержек при реализации адаптеров и ряда других факторов, которые можно учесть в локальной сети.

Адаптер АИС обеспечивает интероперабельное взаимодействие с Подсистемой взаимодействия ИСЭАР. Подсистема взаимодействия поддерживает протоком SOAP для интеграции с адаптерами внешних АИС и специфические протоколы для взаимодействия с облегчёнными адаптерами АИС локальной сети.

Сообщения, принимаемые подсистемой взаимодействия, поступают в подсистему обработки сообщений. Эта подсистема обеспечивает надёжное хранение очереди

сообщений и передачу сообщений деловым процессам в соответствие с их подпиской на специфические типы сообщений.

Подсистема деловых процессов обеспечивает запуск и интерпретацию регламентов, и взаимодействует с подсистемой обработки сообщений для отправки сообщений в различные АИС и получения новых сообщений из АИС.

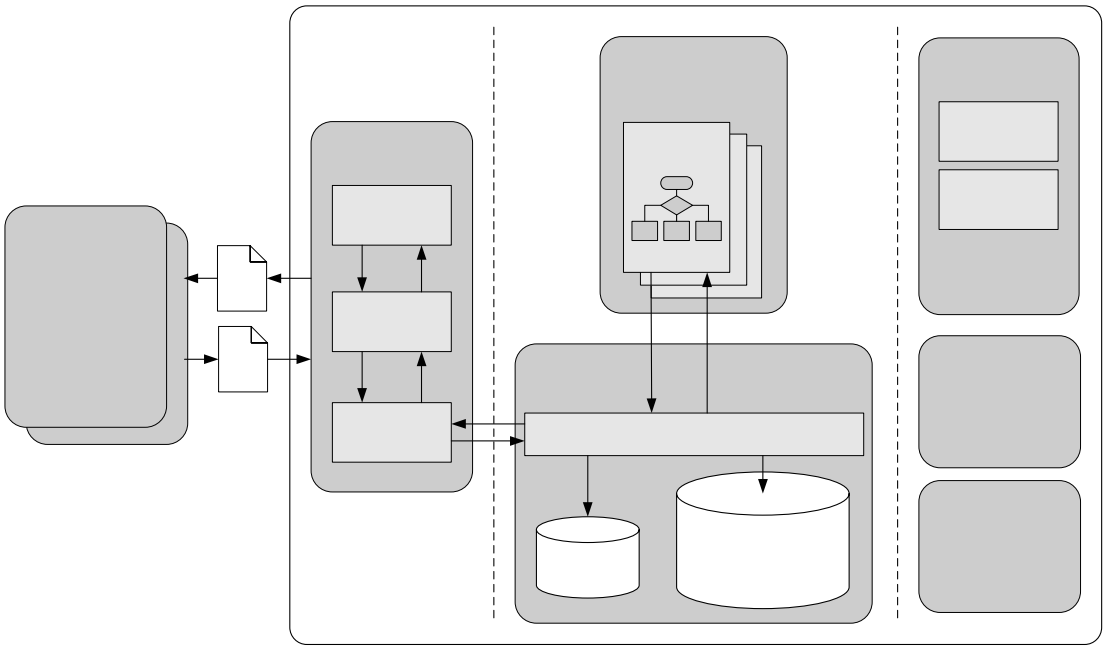


Рисунок 65. Схема функциональной структуры

9.1.4Описание административных сервисов ИС ЭАР

Адаптеры АИС могут использовать ряд административных сервисов ИС ЭАР, реализуемых подсистемой НСИ ИС ЭАР. В их состав входят следующие административные сервисы:

- Сервис отображения глобальных и локальных идентификаторов НСИ ИС ЭАР
- Сервис репликации данных НСИ ИС ЭАР
- Сервис нотификации АИС

9.1.4.1Сервис отображения глобальных и локальных идентификаторов НСИ ИС ЭАР

Сервис отображения глобальных и локальных идентификаторов НСИ ИС ЭАР обеспечивает механизм глобальной идентификации. Интегрируемые АИС, как правило, используют механизмы локальной идентификации своих информационных ресурсов, такие как первичные ключи таблиц БД и пр. В целях обеспечения репликации данных с НСИ и обмена данными в рамках ИСЭАР необходимо глобально идентифицировать информационные ресурсы. Сервис отображения глобальных и локальных

идентификаторов НСИ ИС ЭАР позволяет АИС зарезервировать глобально-уникальные URI-идентификаторы для собственных ресурсов и обеспечить отображение локальных идентификаторов на глобальные.

WSDL-описание и документация к интерфейсу отображения глобальных и локальных идентификаторов НСИ ИС ЭАР приведены в приложениях.

В данном разделе рассматриваются операции, предоставляемые сервисом.

9.1.4.1.1 Map

```
POST /NSIWebService/Mapping.asmx HTTP/1.1
Content-Type: text/xml; charset=utf-8
Content-Length: length

<?xml version="1.0" encoding="utf-8"?>
<soap:Envelope xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xmlns:xsd="http://www.w3.org/2001/XMLSchema"
xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/">
  <soap:Body>
    <Map xmlns="http://www.ultimeta.ru">
      <locIdMappings systemId="string" xmlns="urn:EAR2004-NSIMapping">
        <mapping id="string" type="string">string</mapping>
        <mapping id="string" type="string">string</mapping>
      </locIdMappings>
    </Map>
  </soap:Body>
</soap:Envelope>

HTTP/1.1 200 OK
Content-Type: text/xml; charset=utf-8
Content-Length: length

<?xml version="1.0" encoding="utf-8"?>
<soap:Envelope xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xmlns:xsd="http://www.w3.org/2001/XMLSchema"
xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/">
  <soap:Body>
    <MapResponse xmlns="http://www.ultimeta.ru">
      <MapResult>int</MapResult>
    </MapResponse>
```

```
  </soap:Body>
</soap:Envelope>
```

Метод регистрирует соответствие между локальным идентификатором и URI

systemId – идентификатор подсистемы (фиксированная строка)

<mapping id="**string**" type="**string**">**string**</mapping>:

- id – локальный идентификатор ресурса в подсистеме
- type – тип ресурса в подсистеме (может быть пустым или отсутствовать, что эквивалентно). Используется при наличии нескольких типов ресурсов с пересекающимися множествами идентификаторов
- <значение тега> -- URI

Возвращает:

- 0 – все соответствия успешно зарегистрированы
- != 0 – возникла ошибка, операция полностью отменена

9.1.4.1.2 Unmap

```
POST /NSIWebService/Mapping.asmx HTTP/1.1
Content-Type: text/xml; charset=utf-8
Content-Length: length

<?xml version="1.0" encoding="utf-8"?>
<soap:Envelope xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xmlns:xsd="http://www.w3.org/2001/XMLSchema"
xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/">
  <soap:Body>
    <Unmap xmlns="http://www.ultimeta.ru">
      <locIdUnmappings systemId="string" xmlns="urn:EAR2004-NSIMapping">
        <mapping id="string" type="string" />
        <mapping id="string" type="string" />
      </locIdUnmappings>
    </Unmap>
  </soap:Body>
</soap:Envelope>



HTTP/1.1 200 OK
Content-Type: text/xml; charset=utf-8
```

```
Content-Length: length

<?xml version="1.0" encoding="utf-8"?>
<soap:Envelope xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xmlns:xsd="http://www.w3.org/2001/XMLSchema"
xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/">
  <soap:Body>
    <UnmapResponse xmlns="http://www.ultimeta.ru">
      <UnmapResult>int</UnmapResult>
    </UnmapResponse>
  </soap:Body>
</soap:Envelope>
```

Метод удаляет регистрацию соответствия между локальным идентификатором и URI

systemId – идентификатор подсистемы (фиксированная строка)

<mapping id="**string**" type="**string**">**string**</mapping>:

- id – локальный идентификатор ресурса в подсистеме
- type – тип ресурса в подсистеме (может быть пустым или отсутствовать, что эквивалентно). Используется при наличии нескольких типов ресурсов с пересекающимися множествами идентификаторов

Возвращает:

- 0 – все соответствия успешно удалены
- не 0 – возникла ошибка, операция полностью отменена

9.1.4.1.3 GetIdentifiers

```
POST /NSIWebService/Mapping.asmx HTTP/1.1
Content-Type: text/xml; charset=utf-8
Content-Length: length

<?xml version="1.0" encoding="utf-8"?>
<soap:Envelope xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xmlns:xsd="http://www.w3.org/2001/XMLSchema"
xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/">
  <soap:Body>
    <GetIdentifiers xmlns="http://www.ultimeta.ru">
      <locIdUris systemId="string" xmlns="urn:EAR2004-NSIMapping">
        <mapping>string</mapping>
```

```
        <mapping>string</mapping>
      </locIdUris>
    </GetIdentifiers>
  </soap:Body>
</soap:Envelope>


HTTP/1.1 200 OK
Content-Type: text/xml; charset=utf-8
Content-Length: length

<?xml version="1.0" encoding="utf-8"?>
<soap:Envelope xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xmlns:xsd="http://www.w3.org/2001/XMLSchema"
xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/">
  <soap:Body>
    <GetIdentifiersResponse xmlns="http://www.ultimeta.ru">
      <GetIdentifiersResult xmlns="urn:EAR2004-NSIMapping">
        <mapping id="string" type="string">string</mapping>
        <mapping id="string" type="string">string</mapping>
      </GetIdentifiersResult>
    </GetIdentifiersResponse>
  </soap:Body>
</soap:Envelope>
```

Метод возвращает локальные идентификаторы, соответствующие заданным URI

systemId – идентификатор подсистемы (фиксированная строка)

<mapping id="**string**" type="**string**">**string**</mapping>:

- id – локальный идентификатор ресурса в подсистеме
- type – тип ресурса в подсистеме (может быть пустым или отсутствовать, что эквивалентно). Используется при наличии нескольких типов ресурсов с пересекающимися множествами идентификаторов
- <значение тега> -- URI

При вызове указывается только значение тега (URI)

Возвращает множество элементов <mapping>, для которых найдено соответствие. Локальный идентификатор и тип возвращаются в атрибутах соответствующего элемента. При отсутствии соответствия элемент не возвращается.

9.1.4.1.4 GetUris

```
POST /NSIWebService/Mapping.asmx HTTP/1.1
Content-Type: text/xml; charset=utf-8
Content-Length: length

<?xml version="1.0" encoding="utf-8"?>
<soap:Envelope xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xmlns:xsd="http://www.w3.org/2001/XMLSchema"
xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/">
  <soap:Body>
    <GetUris xmlns="http://www.ultimeta.ru">
      <locIdentifiers systemId="string" createUri="boolean" xmlns="urn:EAR2004-
NSIMapping">
        <mapping id="string" type="string" />
        <mapping id="string" type="string" />
      </locIdentifiers>
    </GetUris>
  </soap:Body>
</soap:Envelope>


HTTP/1.1 200 OK
Content-Type: text/xml; charset=utf-8
Content-Length: length

<?xml version="1.0" encoding="utf-8"?>
<soap:Envelope xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xmlns:xsd="http://www.w3.org/2001/XMLSchema"
xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/">
  <soap:Body>
    <GetUrisResponse xmlns="http://www.ultimeta.ru">
      <GetUrisResult systemId="string" xmlns="urn:EAR2004-NSIMapping">
        <mapping id="string" type="string">string</mapping>
        <mapping id="string" type="string">string</mapping>
```

```
      </GetUrisResult>
    </GetUrisResponse>
  </soap:Body>
</soap:Envelope>
```

Метод возвращает URI, соответствующие заданным локальным идентификаторам.

systemId – идентификатор подсистемы (фиксированная строка)

createUri – значение “true” означает, что при отсутствии URI для локального идентификатора НСИ должна создать URI и зарегистрировать соответствие между ним и локальным идентификатором

<mapping id="**string**" type="**string**">**string**</mapping>:

- id – локальный идентификатор ресурса в подсистеме
- type – тип ресурса в подсистеме (может быть пустым или отсутствовать, что эквивалентно). Используется при наличии нескольких типов ресурсов с пересекающимися множествами идентификаторов
- <значение тега> -- URI

При вызове указываеются локальный идентификатор и (возможно) тип. Значение элемента – пустое

Возвращает множество элементов <mapping>, для которых найдено (или зарегистрировано) соответствие. URI возвращаются в значении элемента. При отсутствии соответствия элемент не возвращается.

9.1.4.2Сервис репликации данных НСИ ИС ЭАР

НСИ ИС ЭАР обеспечивает хранение справочной информации, такой как контролируемые словари и классификаторы, ведомственные организационные справочники и пр. Структуры данных, стандартизованные и хранимые в НСИ в рамках работ 2004 года рассмотрены в разделах 7.10 «Описание базовых схем данных для сквозных пилотных проектов ИС ЭАР» и 7.11 «Контролируемые словари в НСИ ИС ЭАР».

Эти общесистемные справочные данные могут предоставляться, поддерживаться и обновляться одной из АИС в ИСЭАР, такой как АИС «Кадры», отвечающей за организационные справочники и информацию о штате сотрудников ведомства. Другие системы могут использовать справочную информацию. Совместное использование этих информационных ресурсов требует наличие механизма репликации данных НСИ с системами ИСЭАР.

Сервис репликации данных НСИ ИС ЭАР обеспечивает этот механизм репликации. Для уведомления АИС, заинтересованных в некоторых данных, об обновлении информации, используется сервис нотификации АИС (см. раздел 9.1.4.3)

WSDL-описание и документация к интерфейсу репликации данных НСИ ИС ЭАР приведены в приложениях.

В данном разделе рассматриваются операции, предоставляемые сервисом.

9.1.4.2.1 GetData

```
POST /NSIWebService/NSIWebService.asmx HTTP/1.1
Content-Type: text/xml; charset=utf-8
Content-Length: length

<?xml version="1.0" encoding="utf-8"?>
<soap:Envelope xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xmlns:xsd="http://www.w3.org/2001/XMLSchema"
xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/">
  <soap:Body>
    <GetData xmlns="http://www.ultimeta.ru">
      <dataRequest systemId="string" timeStamp="string" xmlns="urn:EAR2004-NSIData
/>
    </GetData>
  </soap:Body>
</soap:Envelope>


HTTP/1.1 200 OK
Content-Type: text/xml; charset=utf-8
Content-Length: length

<?xml version="1.0" encoding="utf-8"?>
<soap:Envelope xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xmlns:xsd="http://www.w3.org/2001/XMLSchema"
xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/">
  <soap:Body>
    <GetDataResponse xmlns="http://www.ultimeta.ru">
      <GetDataResult hash="string" timeStamp="string" xmlns="urn:EAR2004-NSIData">
        <data xmlns="">xml</data>
      </GetDataResult>
```

```
    </GetDataResponse>
  </soap:Body>
</soap:Envelope>
```

Метод используется подсистемами для получения данных из НСИ.

На входе передаются параметры:

- systemId – идентификатор подсистемы
- timeStamp – метка времени (строка, непрозрачная для подсистемы)

На выходе:

- hash – MD5-хеш значения элемента <data> (только внутренний XML)
- timeStamp – новая метка времени
- <data> -- данные в соответствии со схемой

9.1.4.2.2 PutData

```
POST /NSIWebService/NSIWebService.asmx HTTP/1.1
Content-Type: text/xml; charset=utf-8
Content-Length: length

<?xml version="1.0" encoding="utf-8"?>
<soap:Envelope xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xmlns:xsd="http://www.w3.org/2001/XMLSchema"
xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/">
  <soap:Body>
    <PutData xmlns="http://www.ultimeta.ru">
      <dataContent systemId="string" hash="string" xmlns="urn:EAR2004-NSIData">
        <data xmlns="">xml</data>
      </dataContent>
    </PutData>
  </soap:Body>
</soap:Envelope>


HTTP/1.1 200 OK
Content-Type: text/xml; charset=utf-8
Content-Length: length

<?xml version="1.0" encoding="utf-8"?>
```

```
<soap:Envelope xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xmlns:xsd="http://www.w3.org/2001/XMLSchema"
xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/">
  <soap:Body>
    <PutDataResponse xmlns="http://www.ultimeta.ru">
      <PutDataResult>int</PutDataResult>
    </PutDataResponse>
  </soap:Body>
</soap:Envelope>
```

Метод используется АИС «Кадры» для загрузки данных в НСИ.

На входе передаются параметры:

- systemId – идентификатор подсистемы
- hash – MD5-хеш значения элемента <data> (только внутренний XML)
- <data> -- данные в соответствии со схемой

На выходе:

- 0 – сообщение принято, данные загружены
- != 0 – возникла ошибка, данные не загружены, необходима повторная передача данных

9.1.4.3Сервис нотификации АИС

Сервис нотификации АИС используется для уведомления АИС, заинтересованных в некоторых данных, об обновлении информации, реплицируемой сервисом репликации данных НСИ ИС ЭАР (см. раздел 9.1.4.2)

Для получения уведомлений, адаптер АИС должен поддержать интерфейс нотификации АИС. При поддержке этого интерфейса на стороне адаптера, сервис нотификации сможет уведомить АИС о поступлении новой порции данных.

WSDL-описание и документация к интерфейсу нотификации АИС приведены в приложениях.

В данном разделе рассматриваются операции, используемые для нотификации АИС об изменениях реплицированных данных в НСИ ИС ЭАР.

9.1.4.3.1 Notify

```
POST /NSIWebService/Notification.asmx HTTP/1.1
Content-Type: text/xml; charset=utf-8
Content-Length: length
```

```
<?xml version="1.0" encoding="utf-8"?>
<soap:Envelope xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xmlns:xsd="http://www.w3.org/2001/XMLSchema"
xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/">
  <soap:Body>
    <Notify xmlns="http://www.ultimeta.ru" />
  </soap:Body>
</soap:Envelope>
```

Метод реализуется на стороне потребителей информации НСИ для получения уведомлений о наличии в базе НСИ обновлений. Метод не имеет параметров и является однонаправленным, т.е. не требует SOAP-ответа.

9.1.5Разработка адаптера АИС

Процесс взаимодействия АИС с ИСЭАР рассмотрен в разделе 9.1.3 «Общая структура интегрируемых с помощью технологии Web-сервисов компонент ИС ЭАР с АИС». В этом процессе интеграции ИСЭАР с АИС ключевую роль играет «адаптер АИС», обеспечивающий интероперабельность АИС в ИСЭАР.

Требования к адаптерам рассмотрены в разделе «7.12.1 Общие требования к правилам формирования адаптеров ИС». Требования различаются для адаптера внешней АИС и адаптера АИС локальной сети.

Адаптеры могут использовать административные сервисы ИС ЭАР, предоставляемые подсистемой НСИ ИС ЭАР. Подробнее административные сервисы рассмотрены в разделе 9.1.4 «Описание административных сервисов ИСЭАР».

ИСЭАР содержит вспомогательные средства поддержки реализации адаптеров, в частности, за счёт средств BizTalk Server 2004, позволяющих организовать вспомогательный рабочий процесс, отвечающий за преобразование форматов и протоколов.

Раздел 9.1.6 «Интеграция ИСЭАР с АИС пилотных проектов» вместе с приложениями содержит содержательный пример средств и способов интеграции ИСЭАР с АИС в пилотных проектах 2004 года.

9.2 Требования к интегрируемой ИС

Основным условием взаимодействия является то, что обмен информации с внешними подсистемами может осуществляться только на базе XML.

Интегрируемая в ИС ЭАР информационная система должна удовлетворять требованиям к поддерживаемым в ИС ЭАР стандартам и рекомендациям по их применению в рамках ИС ЭАР, описанным в разделе «8. Рекомендации по использованию стандартов в рамках ИС ЭАР».

Поддержка АИС этих требований обеспечивается путём реализации «адаптера АИС», обеспечивающего её интеграцию в ИСЭАР. В частности, адаптер внешней ИС предоставляет интерфейс веб-сервиса к АИС, обеспечивает поддержку единой модели данных и стандартных схем данных, регламентированных ИСЭАР. Подробнее про адаптеры в ИСЭАР см. раздел «8.12.1 Общие требования к правилам формирования адаптеров ИС».

Помимо этого:

- Перед интеграцией с АИС ее атрибуты, предоставляемые сервисы, необходимо прописать в реестре АИС исполняющей системы ЭАР.
- Вероятно, потребуется создать добавить новые схемы документов и зарегистрировать их в Реестре схем документов. XML-схема должна быть согласована с общими стандартными схемами данных, зарегистрированными в Репозитории схем документов ИС ЭАР. Специализированная схема должна быть организована как расширение базовой схемы.
- для классификации своих услуг и данных пользоваться стандартными классификаторами и средствами НСИ ИС ЭАР, описанными в разделе 9.1.

9.3 Организация работ по проведению интеграции ИС с ИС ЭАР

Разработкой новых деловых процессов и подключением новых подсистем занимается разработчик процессов, который должен обладать следующими знаниями и навыками:

- Опытом визуальной разработки деловых процессов в выбранной среде проектирования
- Опытом проектирования web-сервисов
- Опытом разработки схем данных и сообщений
- Знакомством с Руководством администратора ИСЭАР

должен обладать хорошим знанием предметной области интегрируемой ИС, достаточными для выделения общих и канонических структур данных на вербальном

уровне. Второй специалист должен обладать хорошими знаниями стандарта W3C XML Schema и владеть технологией XSLT для формирования преобразователей данных.

Приблизительные трудозатраты на весь процесс приведения форматов данных ИС к форматам данных СЭВ – 20 чел/дней для каждого из участников процесса.

Для обмена данными между ИС ЭАР и АИС в общем случае должны быть пройдены следующие этапы (хотя в частных случаях АИС может уже удовлетворять части требований, соответственно по некоторым этапам проведение работ не понадобится)

9.3.1 Выявление услуг, предоставляемых и получаемых АИС посредством ИСЭАР, спецификация прецедентов взаимодействия АИС и способа интеграции в ИСЭАР

На этом этапе производится выявление функциональных требований к интеграции АИС в ИСЭАР. Производится выявление АИС и подсистем ИСЭАР, с которыми должна взаимодействовать АИС, выявление услуг, которые данная АИС должна предоставлять другим АИС в контексте их взаимодействия, и услуг, которые данная АИС намерена получать от других АИС посредством ИСЭАР.

Также специфицируются прецеденты взаимодействия АИС с ИСЭАР, АИС с другими АИС. Анализируются и документируются ЭАРы, в которых будет участвовать интегрируемая АИС.

Кроме того, необходимо произвести выбор способа интеграции АИС в ИСЭАР – типа «адаптера АИС» и средств его описания (см. раздел 8.12 «Адаптеры АИС»). Если АИС является внешней по отношению к ведомственной сети, в которой работает ИСЭАР, то адаптер АИС должен удовлетворять требованиям «внешнего адаптера», описанным в разделе 8.12.1 «Общие требования к правилам формирования адаптеров ИС» и в целом в главах 4, 5 и 7. То есть, адаптер должен следовать архитектуре WSA и использовать SOAP для транспорта и XML для представления данных. Если же АИС функционирует исключительно в локальной сети, то требования существенно менее строгие. Для реализации как первого, так и второго типа адаптера могут использоваться средства Подсистемы разработки адаптеров ИСЭАР (см. раздел 9.1.1.3 «Состав ИСЭАР»), для реализации адаптера внешней АИС могут использоваться любые другие инструментальные средства.

Результатом должна являться спецификация функциональных требований к взаимодействию АИС в ИСЭАР и другими АИС, спецификация прецедентов взаимодействия, спецификация ЭАР, в которых будет участвовать данная АИС.

Результаты могут быть проиллюстрированы UML-диаграммами прецедентов, последовательности, взаимодействия, деятельности.

Требования к квалификации персонала: выявлением функциональных требований и прецедентов взаимодействия должен заниматься бизнес-аналитик. Аналитик должен обладать хорошим знанием предметной области интегрируемой ИС и требований к взаимодействию её с ИСЭАР и другими АИС.

Приблизительные трудозатраты: 5 дней/чел.

9.3.2 Представление данных АИС в виде XML

Существенным условием является то, что в процессе исполнения ЭАР все требуемые в процессе данные должны быть представимы в XML.

В случае выбора механизма «внешнего адаптера» для интеграции АИС в ИСЭАР, обмен информацией с АИС может осуществляться только на базе XML. Для упрощения этого шага могут использоваться средства Подсистемы разработки адаптеров ИСЭАР или, в случае выбора механизма «внутреннего адаптера», встроенные транспортные протоколы и конвейеры разбора данных Подсистемы взаимодействия ИСЭАР (см. раздел 9.1.1.3 «Состав ИСЭАР»), в частности, обеспечиваемые используемым в рамках работ 2004 года сервером интеграции (Microsoft BizTalk Server 2004).

Соответственно, если система в исходном варианте не поддерживает этой возможности, должны быть проведены работы по настройке возможности XML-отображения для данных АИС (конвертации данных в XML и обратно).

Требования к квалификации персонала: в случае реализации на стороне АИС - технический специалист, разбирающихся в деталях внутренней инфраструктуры АИС. В случае выбора способа реализации на основе Подсистемы разработки адаптеров ИСЭАР или Подсистемы взаимодействия ИСЭАР - опытом визуальной разработки деловых процессов и конвейеров, схем данных и сообщений в выбранной среде проектирования (Microsoft BizTalk Server 2004) и знакомством с Руководством администратора ИСЭАР

Приблизительные трудозатраты: 5 дней/чел.

9.3.3Описание и согласование XML-схемы данных АИС

Должна быть специфицирована XML-схема документов на языке XML Schema. Для разработки схемы может использоваться редактор схем ИС ЭАР, либо инструментарий UML-моделирования и генератор XML-схем в соответствие с общими рекомендациями раздела 5.3 «Каноническая модель данных и язык спецификации схем данных». XML-схема должна быть согласована с общими стандартными схемами данных,

зарегистрированными в Реестре XML-схем ИС ЭАР. Специализированная схема должна быть организована как расширение базовой схемы.

Требования к квалификации персонала: аналитик, разбирающийся в схеме данных АИС, прецедентах её взаимодействия и общими стандартными схемами данных НСИ ИС ЭАР, а также технический специалист, обладающий опытом разработки XML-схем данных в выбранном визуальном инструментарии (в частности, Подсистемы разработки XML-схем ИСЭАР).

Приблизительные трудозатраты: 3 дня/чел.

9.3.4Регистрация схем данных АИС в Реестре схем документов ИС ЭАР

XML-схемы документов, передаваемых АИС, должны быть зарегистрированы в Реестре схем документов ИС ЭАР. Результатом этапа являются зарегистрированные в ИС ЭАР XML-схемы.

Требования к квалификации персонала: оператор, знакомый с Руководством администратора ИСЭАР.

Приблизительные трудозатраты: 1 день/чел.

9.3.5Этап 4. Реализация «адаптера АИС»

Для обмена данными между ИС ЭАР и ИС должен быть соответствующим способом настроен транспортный уровень – подсистема взаимодействия ИС ЭАР. На этом этапе происходит построение адаптера ИС, позволяющего выполнять обмен данными АИС и ИС ЭАР.

Тип адаптера и способ его реализации определяется выбором, сделанным аналитиком на Этапе 1 (см. раздел 9.3.1).

Результатом данного этапа является созданный адаптер ИС, поддерживающий стандартный интерфейс и позволяющий осуществить интеграцию с ИС ЭАР.

Требования к квалификации персонала:

- в случае реализации «адаптера внешней АИС» на основе Подсистемы разработки адаптеров ИСЭАР – технический специалист, обладающий опытом проектирования web-сервисов, визуальной разработки деловых процессов и конвейеров в среде проектирования Microsoft BizTalk Server 2004 и знакомством с Руководством администратора ИСЭАР.

- в случае реализации «адаптера внешней АИС» на основе стороннего инструментария - технический специалист, обладающий опытом проектирования и разработки web-сервисов в этом инструментарии.
- в случае реализации «адаптера АИС локальной сети» средствами Подсистемы взаимодействия ИСЭАР – технический специалист, обладающий опытом визуальной разработки деловых процессов и конвейеров в среде проектирования Microsoft BizTalk Server 2004 и знакомством с Руководством администратора ИСЭАР.

Приблизительные трудозатраты: 10 дней/чел.

9.3.6Регистрация АИС в Реестре АИС ИС ЭАР

Для завершения интеграции с АИС ее атрибуты необходимо прописать в реестре АИС исполняющей системы ЭАР, см. раздел 9.1. Результатом данного этапа является готовность АИС к использованию в деловых процессах ЭАР.

Требования к квалификации персонала: оператор, знакомый с Руководством администратора ИСЭАР.

Приблизительные трудозатраты: 1 день/чел.

10 Интеграция СЭВ с ИС ЭАР

В данном разделе приводятся требования к оформлению адаптеров ЭАР ИС ЭАР для возможности их интеграции в рамках процедур электронных взаимодействий СЭВ. Регламентируется процесс регистрации схем данных ИС ЭАР в СЭВ, определяются требования к поддержке единых механизмов безопасности и классификации сервисов и услуг.

Рассматриваются этапы работ по интеграции ИС ЭАР в СЭВ с описанием квалификации выполняющих интеграцию сотрудников и приблизительных трудозатрат.

10.1 Описание требований к СЭВ со стороны ИС ЭАР

Поскольку система внутриведомственной интеграции ИС ЭАР взаимодействует с СЭВ как одна из систем, интегрируемых в рамках межведомственного взаимодействия, то первичными требованиями по интеграции этих двух систем являются требования к ИС ЭАР со стороны СЭВ. В свою очередь ИС ЭАР должен полностью соответствовать данным требованиям для возможности взаимодействия с СЭВ.

К основным требованиям к СЭВ со стороны ИС ЭАР является:

- следование определенному в данном документе набору стандартов WSA;
- возможность использования СЭВ в качестве транспортной среды ИС ЭАР при межведомственном взаимодействии, что полностью покрывается возможностями СЭВ по передачи сообщений по протоколам HTTP/HTTPS.

10.2 Организация работ по проведению интеграции ИС ЭАР с СЭВ

При выполнении работ по интеграции ИС ЭАР с СЭВ должны быть учтены требования к ИС ЭАР со стороны СЭВ, которые приведены в документе «Создание комплекта проектов основных технических нормативных документов по взаимодействию Среды с внешними системами и ИСЭАР» в разделе «8.1 Описание требований к ИС ЭАР со стороны СЭВ», разработанном в рамках государственного контракта № ЭР.03.16 от 21.07.04 «Развитие процедур электронного взаимодействия органов государственной власти и хозяйствующих субъектов».

10.2.1 Этап 1. Регистрация канонических схем данных ИС ЭАР в СЭВ

На данном этапе интеграции необходимо выделить базовые структуры данных ИС ЭАР, выразить их в терминах стандартных схем СЭВ и зарегистрировать в хранилище метаданных СЭВ. Работы по данному этапу разбиваются на следующую последовательность действий:

10.2.1.1 Выделение базовых структур данных

В рамках данных работ необходимо провести анализ существующих данных, размещаемых в хранилищах данных и взаимодействующих в рамках ИС ЭАР информационных систем, а так же данных, хранимых непосредственно в репозитариях ИС ЭАР (например, справочники НСИ). После этого необходимо провести выделение объектной схемы ресурсов пилотных проектов ИС ЭАР на базе проведенного анализа. Результаты преобразования существующей модели данных к объектной необходимо представить в виде набора XML-схем, описывающих информационные сущности ИСЭАР.

10.2.1.2 Построение преобразователей от общих к каноническим схемам данных

После выделения общих схем ресурсов взаимодействующих в рамках ИС ЭАР информационных систем, необходимо построить подсхемы, общие для всех интегрируемых информационных областей – канонические схемы данных. Так же необходимо реализовать XSLT-преобразователи данных от общих схем к каноническим.

10.2.1.3 Отображение канонических схем данных ИС ЭАР и СЭВ

Далее необходимо выполнить отображение полученных таким образом канонических (стандартных) схем данных ИС ЭАР на канонические (стандартные) схемы данных СЭВ. При этом в качестве результата надо определить набор XSLT-преобразователей данных от канонических схем ИСЭАР к каноническим схемам СЭВ.

В итоге необходимо зарегистрировать общие XML-схемы данных ИСЭАР, канонические схемы данных ИСЭАР, преобразователи между общими и каноническими схемами ИСЭАР и преобразователи между каноническими схемами ИСЭАР и СЭВ в репозитарии схем данных СЭВ, что позволит автоматически в два этапа выполнять преобразование данных ИСЭАР к каноническим форматам данных СЭВ.

10.2.1.4 Требования к квалификации сотрудников и приблизительные трудозатраты

Для выполнения указанных работ требуются два специалиста. Первый из них – эксперт – должен обладать хорошим знанием предметных областей пилотных проектов ИС ЭАР, достаточными для выделения общих и канонических структур данных на вербальном уровне. Второй специалист должен обладать хорошими знаниями стандарта W3C XML Schema и владеть технологией XSLT для формирования преобразователей данных.

Приблизительные трудозатраты на весь процесс приведения форматов данных ИС ЭАР к форматам данных СЭВ – 1 чел/месяц для каждого из участников процесса.

10.2.2 Этап 2. Реализация внешних адаптеров ИС

На данном этапе для каждого из пилотных проектов ИСЭАР (ЭАР-ов) необходимо реализовать внешний адаптер к СЭВ – Web-сервис, позволяющий осуществлять взаимодействие с ЭАР-ом из Процедур Электронных Взаимодействий СЭВ.

10.2.2.1 Поддержка требований к внутренним сервисам СЭВ

Поскольку платформой реализации ИС ЭАР является Microsoft Biztalk 2004, то возможность оформления ЭАР-ов как Web-сервисов уже поддерживается средствами данной платформы. Между тем необходимо привести интерфейс данного сервиса для каждого ЭАР-а к интерфейсу, удовлетворяющему требованиям к оформлению интерфейсов внутренних сервисов СЭВ, описанным в разделе «7.2 Требования к интегрируемой ИС». Данные доработки предполагают расширение WSDL-интерфейса ЭАР-а дополнительными операциями, используемыми для взаимодействия с сервисом исполнения ПЭВ СЭВ.

10.2.2.2 Поддержка стандартов взаимодействия СЭВ

Для взаимодействия с сервисом исполнения ПЭВ СЭВ Web-сервис ЭАР-а должен поддерживать протоколы обмена SOAP-сообщениями СЭВ и реализовывать правильную модель оформления удаленных вызовов. В связи с этим необходимо доработать адаптеры ИСЭАР так, чтобы они:

- поддерживали инкапсуляцию SOAP-сообщений в HTTP/HTTPS протоколы с методом POST-привязки;
- поддерживали стандартную модель RPC-стиля SOAP-сообщений.

10.2.2.3 Регистрация адаптеров ИС ЭАР в UDDI реестре СЭВ

После реализации адаптеров ИСЭАР необходимо произвести их регистрацию в UDDI реестре СЭВ, определив для них стандартные классификаторы подсистемы классификации СЭВ.

10.2.2.4 Требования к квалификации сотрудников и приблизительные трудозатраты

Для выполнения указанных работ требуются два специалиста. Первый из них – эксперт – должен иметь хорошее знание назначения ЭАР-ов ИСЭАР для привязки их к системе классификации услуг UDDI реестра СЭВ. Второй специалист должен обладать:

- хорошими знаниями стандартов W3C SOAP, W3C WSDL;
- умением разрабатывать адаптеры на платформе Microsoft Biztalk 2004.

Приблизительные трудозатраты на весь процесс реализации адаптеров ИС ЭАР – 1,5 чел/месяц технического специалиста и 10 чел/дней эксперта.

10.2.3 Этап 3. Интеграция с подсистемой безопасности СЭВ

На данном этапе необходимо обеспечить интеграцию подсистемы информационной безопасности ИС ЭАР и подсистемы информационной безопасности СЭВ.

10.2.3.1 Согласование ролей пользователей СЭВ и ИСЭАР

При вызове ЭАР-ов из ПЭВ ему передается идентификатор пользовательской сессии СЭВ. Адаптер ИСЭАР должен обращаться к сервису авторизации СЭВ с полученным идентификатором сессии и получать список ролей пользователя.

Далее необходимо выполнять отображение ролей пользователя СЭВ на роли пользователя ИС ЭАР и, тем самым, производить авторизацию пользователя СЭВ в ИС ЭАР.

10.2.3.2 Требования к квалификации сотрудников и приблизительные трудозатраты

Для выполнения указанных работ требуется технический специалист, знакомый с интерфейсами сервиса авторизации и структурой пользовательских ролей СЭВ, структурой пользовательских ролей ИСЭАР.

Приблизительные трудозатраты на процесс интеграции подсистем безопасности – 0,5 чел/месяцев технического специалиста.

10.2.4 Этап 4. Интеграция с подсистемой классификаторов СЭВ

На данном этапе необходимо произвести интеграцию систем классификации СЭВи ИСЭАР.

10.2.4.1 Загрузка ключевых классификаторов ИС ЭАР в подсистему классификаторов СЭВ

Для возможности публикации данных по ЭАР-ам и участвующим в них Web-сервисам в СЭВ необходимо подготовить и загрузить ключевые классификаторы ИС ЭАР в подсистему управления классификаторами СЭВ.

10.2.4.2 Регистрация внутренних сервисов ИС ЭАР в UDDI реестре СЭВ

Используя загруженные классификаторы ИС ЭАР, необходимо обеспечить возможность регистрации сервисов-участников ЭАР-ов в UDDI реестре СЭВ с привязкой как к загруженным классификаторам ИС ЭАР, так и к унифицированным классификаторам СЭВ для возможности поиска информации по предоставляемым услугам как в разрезе специализированных, так и универсальных рубрикаторов СЭВ.

10.2.4.3 Требования к квалификации сотрудников и приблизительные трудозатраты

Для выполнения указанных работ требуются два специалиста. Первый из них – эксперт – должен иметь хорошее знание структуры классификаторов СЭВ и ИСЭАР. Второй специалист должен обладать:

- хорошими знаниями технологии XSLT для преобразования выгруженных в XML-формате классификаторов ИСЭАР к формату СЭВ;
- знанием интерфейсов подсистемы классификаторов СЭВ для возможности загрузки преобразованных классификаторов ИСЭАР.

Приблизительные трудозатраты на весь процесс интеграции с подсистемой классификаторов СЭВ – 1 чел/месяц эксперта и 0,5 чел/месяца технического специалиста.

10.3 Описание требований к ИС ЭАР со стороны СЭВ.

Информационное взаимодействие СЭВ с ИС ЭАР осуществляется на уровне оркестрирования внутриведомственных электронных административных регламентов с помощью процедур электронных взаимодействий. В связи с этим, поскольку каждый ЭАР в качестве внешнего интерфейса представлен Web-сервисом, то задача интеграции ИС ЭАР в СЭВ пересекается с задачей интеграции произвольной ИС в СЭВ. Помимо общих требований к ИС ЭАР как к интегрируемой в СЭВ системе, существует и ряд специфичных требований, определяемых важной ролью ИС ЭАР в задаче интеграции внутриведомственных ИС.

Для возможности использования ЭАР-ов, предоставляемых ИС ЭАР в качестве участников ПЭВ необходимо выполнение базовых требований к ИСЭАР со стороны СЭВ.

10.3.1 Требования к схеме данных ИС ЭАР

Для обеспечения информационной интероперабельности необходимо выполнение следующих требований к регистрации схем данных ИС ЭАР:

- необходимость регистрации канонических схем данных и схем данных конкретных внутриведомственных сервисов в репозитарии схем данных СЭВ;
- необходимость регистрации преобразователей схем данных внутриведомственных сервисов к каноническим и обратно;
- необходимость регистрации преобразователей типовых (канонических) схем данных ИСЭАР к типовым (каноническим) схемам данных СЭВ и обратно;
- взаимодействие с модулем репозитария ПЭВ для прямого и обратного преобразования данных внутриведомственных сервисов ИС ЭАР к типовым данным СЭВ.

10.3.2 Требования к реализации адаптеров ЭАР-ов

Для обеспечения интероперабельности на уровне адаптеров ИСЭР необходимо выполнение следующих требований:

- адаптеры ИС ЭАР (Web-сервисы для взаимодействия с ЭАР-ами) должны быть реализованы в соответствие с требованиями по оформлению интерфейсов внутренних Web-сервисов СЭВ, описанных в разделе «7.2 Требования к интегрируемой ИС»;
- все адаптеры ИС ЭАР (Web-сервисы для взаимодействия с ЭАР-ами) должны быть доступны в UDDI реестре СЭВ и классифицированы в соответствии с отражаемой ими тематикой внутриведомственных взаимодействий;

- версии стандартов для описания интерфейсов и протоколов взаимодействия с адаптерами ИС ЭАР должны быть согласованы с версиями этих стандартов, поддерживаемыми СЭВ, в соответствии с профилем стандартов WS-I Basic Profile 1.0.

10.3.3 Требования к надежности и безопасности

В связи с повышенной ролью интегрируемых в СЭВ сервисов ИС ЭАР в общей инфраструктуре межведомственных взаимодействий к ним предъявляются следующие требования по надежности и безопасности:

- бизнес-процессы, реализующие ЭАР-ы должны поддерживать базовые средства реализации долгоживущих бизнес-транзакций и обработки исключительных ситуаций на уровне используемого языка описания рабочих процессов;
- адаптеры ИС ЭАР должны иметь возможность выполнять привязку передаваемых SOAP-сообщений к защищенному протоколу HTTPS;
- для аутентификации и авторизации обращения пользователей СЭВ к ЭАР-ам в рамках процедур электронных взаимодействий адаптеры ИС ЭАР должны использовать сервис авторизации, предоставляемый модулем авторизации СЭВ.

10.3.4 Требования к стандартизации

Для обеспечения стандартизации и регламентации используемых систем классификации участников взаимодействия, услуг и сервисов необходимо выполнение следующего ряда требований к ведению словарей ИС ЭАР:

- ключевые классификаторы ведомств, их услуг и сервисов ИС ЭАР должны быть приведены к формату СЭВ и загружены в подсистему управления классификаторами ССК СЭВ;
- для классификации и поиска по рубрикатору ведомств, сервисов и услуг в рамках ИС ЭАР должно происходить взаимодействие с модулем классификаторов СЭВ для просмотра данных в разрезе унифицированных в рамках всей рубрикаторов;
- при регистрации сервисов-участников ЭАР в реестре услуг ИС ЭАР должна быть обеспечена возможность регистрации их и в UDDI реестре СЭВ с привязкой к унифицированным классификаторам СЭВ.

10.4 Организация работ по проведению интеграции СЭВ с ИС ЭАР

При выполнении работ по интеграции СЭВ с ИС ЭАР должны быть учтены требования к СЭВ со стороны ИС ЭАР, которые приведены в документе «Разработка комплекта проектов основных технических нормативных документов, определяющих принципы технического взаимодействия ИСЭАР (исполняющей системы электронных административных регламентов) с другими информационными системами, комплекта проектов основных технических нормативных документов, определяющих принципы технического взаимодействия ИСЭАР с СЭВ (системой электронного взаимодействия) (со стороны ИСЭАР)» в разделе «7.1 Описание требований к СЭВ со стороны ИС ЭАР», разработанном в рамках государственного контракта № ЭР.03.17 от 09.06.04 «Разработка и внедрение электронных административных регламентов на примере структурных подразделений Минэкономразвития России».

Дополнительные работы по удовлетворению требований ИС ЭАР к СЭВ в данном документе не регламентируются, поскольку все необходимые механизмы интеграции со стороны СЭВ присутствуют в текущей ее реализации.

11 Заключение

Автоматизированные системы управления взаимодействиями – это не просто инновационные решения и технологии, позволяющие объединить и качественно расширить функционирование всех систем министерства, реализуемые СУВ (АИС «ИСЭАР» и АИС «СЭВ») позволяют интегрировать существующие направления реформы государственного управления: административную реформу, реформу государственной службы и построение «электронного Министерства» и «электронного Государства». С помощью АИС «ИСЭАР» и АИС «СЭВ» реализуется главная идея административной реформы – оптимизация структуры и функций органов государственной власти с ориентацией на результат деятельности. Они ориентированы на решение задач, поставленных перед реформой государственной службы. В частности, за счет внедрения электронной регламентации средствами АИС «ИСЭАР» и АИС «СЭВ» реализуются многие цели, поставленные в Концепции реформирования государственной службы, например:

- идея служебного регламента
- повышение открытости государственной службы
- оптимизация модели поступления и прохождения государственной службы
- повышение эффективности механизма профилактики коррупции и борьбы с ней.

АИС «ИСЭАР» и АИС «СЭВ” является реальными интегрирующими механизмами, которые объединяют воедино элементы «электронного Министерства» и «электронного Государства». Для технологичного решения поставленных задач предложен обоснованный комплексный интеграционный подход, включающий:

- практические принципы, требования и методики реализации автоматизированных систем управления взаимодействиями;
- практические принципы, требования и методики реализации интеграции распределенных, гетерогенных АИС и приложений, реализации электронных административных регламентов и процедур электронного взаимодействия;
- программные решения - АИС «ИСЭАР» и АИС «СЭВ», позволяющие объединить существующие информационные ресурсы и системы;
- применение этих интеграционных решений в реализации электронных административных регламентов и процедур электронного взаимодействия.

12 Источники разработки

При разработке технической нормативной документации использовались следующие источники:

1. [XML] Extensible Markup Language (XML) 1.0; World Wide Web Consortium Recommendation; http://www.w3.org/TR/REC-xml.
2. [RDF] Resource Description Framework (RDF): Concepts and Abstract Syntax, W3C Recommendation, Graham Klyne, Jeremy Carroll, eds. // http://www.w3.org/TR/rdf-concepts/;
3. [RDFS] Resource Description Framework (RDF) Schema Specification // http://www.w3.org/TR/2000/CR-rdf-schema-20000327; http://www.w3.org/TR/1998/WD-rdf-schema
4. [Unicode] The Unicode Standard.
5. [URI] T. Berners-Lee, R. Fielding, L. Masinter. “Uniform Resource Identifiers (URI): Generic Syntax”, IETF RFC 2396, August 1998.
6. [RFC1737] Sollins, K., and L. Masinter. "Functional Requirements for Uniform Resource Names", RFC 1737, MIT/LCS, Xerox Corporation, December 1994.
7. [RFC1738] Berners-Lee, T., Masinter, L., and M. McCahill, Editors. "Uniform Resource Locators (URL)", RFC 1738, CERN, Xerox Corporation, University of Minnesota, December 1994.
8. [NS] Tim Bray, Dave Hollander, and Andrew Layman. Namespaces in XML. World Wide Web Consortium.14-January-1999 http://www.w3.org/TR/REC-xml-names
9. [HDL] The Handle System Home Page, http://www.handle.net/
10. [DOI] International DOI Foundation web site; http://www.doi.org
11. [ISO 11179] ISO/IEC 11179, Specification and Standardization of Data Elements; ftp://sdct-sunsrv1.ncsl.nist.gov/x3l8/11179.
12. [DC] Dublin Core Metadata Initiative, http://dublincore.org/
13. [WF] Lagoze, Carl, Clifford A. Lynch, and Ron Daniel Jr., The Warwick Framework: A Container Architecture for Aggregating Sets of Metadata, Cornell Computer Science Technical Report TR96-1593, July 1996, http://cs-tr.cs.cornell.edu:80/Dienst/UI/2.0/Describe/ncstrl. ornell/TR96-1593
14. [vCard] vCard: The Electronic Business Card, http://www.imc.org/pdi/
15. [OMG] Object Management Group. // http://www.omg.org

16. [CORBA] Common Object Request Brocker Architecture. Version 2.3, OMG Document ad/99-07-01, http://www.omg.org/corba/
17. [UML]: M. Page-Jones and L.L. Constantine, Fundamentals of Object-Oriented Design in UML, Addison-Wesley-Longman, Reading, Mass., 1999]
18. [XMI] XML Metadata Interchange, ftp://ftp.omg.org/pub/docs/ad/98-10-05.pdf
19. [JMS] Java Messaging Service. http://java.sun.com/products/jms/
20. [SW] Semantic Web Activity. // http://www.w3.org/2001/sw
21. [RDF Primer] RDF Primer. W3C Working Draft. // http://www.w3.org/TR/rdf-primer
22. [RDF/XML] RDF/XML Syntax Specification (Revised). W3C Working Draft. // http://www.w3.org/TR/rdf-syntax-grammar/
23. [AGLS] National Archives of Australia – AGLS. // http://www.naa.gov.au/recordkeeping/gov_online/agls/summary.html
24. [OWL] OWL Web Ontology Language 1.0 Reference. W3C Working Draft. // http://www.w3.org/TR/owl-ref/
25. [UDDI] Universal Description, Discovery and Integration specification. // http://www.uddi.org
26. [WSDL] Web Services Description Language. // http://msdn.microsoft.com/xml/general/wsdl.asp
27. [WFMC] Workflow Management Coalition standards. // http://www.wfmc.org/standards/standards.htm
28. [XPDL] Workflow Process Definition Interface – XML Process Definition Language. // http://www.wfmc.org/standards/TC-1025_10_xpdl_102502.pdf
29. [WSFL] Web Services Flow Language. // http://xml.coverpages.org/wsfl.html
30. [XLANG] Web Services for Business Process Design. // http://xml.coverpages.org/xlang.html
31. Transactions in the world of Web services. // http://www-106.ibm.com/developerworks/webservices/library/ws-autobp/?dwzone=webservices
32. Automating business processes and transactions in Web services. // http://www-106.ibm.com/developerworks/webservices/library/ws-autobp/?dwzone=webservices
33. [WS-TX] Web Services Transaction (WS-Transaction). // http://www-106.ibm.com/developerworks/webservices/library/ws-transpec/
34. [WS-C] Web Services Coordination (WS-Coordination). // http://www-106.ibm.com/developerworks/library/ws-coor/
35. [BPML] BPML working draft March 25, 2002. // http://xml.coverpages.org/bpml.html

36. A comparative analysis of WS-Coordination/WS-Transaction and OASIS BTP. // http://www-106.ibm.com/developerworks/webservices/library/ws-comproto/
37. [BTP] Business Transaction Protocol (BTP). // http://www.oasis-open.org/committees/download.php/9836/business_transaction-btp-1.1-spec-wd-04.pdf
38. [BPEL4WS] Business Process Execution Language for Web Services Version 1.1. // http://www-106.ibm.com/developerworks/library/ws-bpel/
39. Robert Shapiro, “A Comparison of XPDL, BPML and BPEL4WS”. // http://xml.coverpages.org/Shapiro-XPDL.pdf
40. [SOAP] SOAP Версия 1.2 Часть 0: Учебник для начинающих. // http://www.w3.org/2002/07/soap-translation/russian/part0.html
41. [WS-CTX] Web Service Context (WS-CTX). // http://www.arjuna.com/library/specs/ws_caf_1-0/WS-CTX.pdf
42. [WS-TXM] Web Service Transaction Management (WS-TXM). // http://www.arjuna.com/library/specs/ws_caf_1-0/WS-TXM.pdf
43. Business processes in a Web services world. // http://www-106.ibm.com/developerworks/library/ws-bpelwp/
44. Schematron // http://www.schematron.com/
45. PRISM: Publishing Requirements for Industry Standard Metadata. http://www.prismstandard.org/
46. А.В. Данилин. Стандарты и международный опыт построения порталов органов государственной власти
47. Н. Игнатович. Брокер интеграции приложений, Открытые системы, #09/2003

13 Приложения

13.1 Приложение. Переводы W3C-стандартов

Приложение представляет начальный набор соответствующих W3C-стандартов.

13.1.1 SOAP 1.1

В отдельном файле (документе) - SOAP_1.1.doc

13.1.2 XML_Схемы_Типы_Данных

В отдельном файле (документе) – XML_Схемы_Типы_Данных.doc

13.1.3 XML_Схемы_Структуры

В отдельном файле (документе) - XML_Схемы_Структуры.doc

13.1.4 WSDL 1.1

В отдельном файле (документе) - WSDL_1.1.doc

13.1.5 BPEL4WS_1.1

В отдельном файле (документе) - BPEL4WS_1.1.doc

13.1.6 UDDI_2.03_Data_Structure

В отдельном файле (документе) - UDDI_2.03_Data_Structure

13.1.7 UDDI _2.04_API_Specification

В отдельном файле (документе) - UDDI _2.04_API_Specification.doc

13.2Приложение. Схемы типовых данных

Приложение описывает схемы типовых данных, сформированные в части работ по реализации подсистемы НСИ АИС «ИСЭАР».

13.2.1 XML-схема орг.справочника НСИ ИСЭАР

Размещается в файле (документе) – СхемыТиповыхДанных.doc

13.2.2 Документация по XML-схеме орг.справочника НСИ ИС ЭАР

Размещается в файле (документе) – СхемыТиповыхДанных.doc

13.2.3 Пример XML-файла с данными по к XML-схеме орг.справочника НСИ ИСЭАР

Размещается в файле (документе) – СхемыТиповыхДанных.doc

13.2.4 XML-схема для представления контролируемых словарей в НСИ ИСЭАР

Размещается в файле (документе) – СхемыТиповыхДанных.doc

13.2.5 Документация по XML-схеме для представления контролируемых словарей в НСИ ИС ЭАР

Размещается в файле (документе) – СхемыТиповыхДанных.doc

13.2.6 Пример XML-представления классификатора в НСИ ИСЭАР

Размещается в файле (документе) – СхемыТиповыхДанных.doc

13.3 Приложение. Спецификация административных интерфейсов ИСЭАР

Приложение описывает административные интерфейсы АИС «ИСЭАР».

13.3.1 WSDL-спецификация интерфейса отображения URI-идентификаторов

Размещается в файле (документе) – АдминистративныеИнтерфейсыИСЭАР.doc

13.3.2 WSDL-спецификация интерфейса репликации данных

Размещается в файле (документе) – АдминистративныеИнтерфейсыИСЭАР.doc

13.3.3 WSDL-спецификация интерфейса нотификации

Размещается в файле (документе) – АдминистративныеИнтерфейсыИСЭАР.doc

13.3.4 Документация к WSDL интерфейсу отображения URI-идентификаторов

Размещается в файле (документе) – АдминистративныеИнтерфейсыИСЭАР.doc

13.3.5 Документация к WSDL интерфейсу репликации данных

Размещается в файле (документе) – АдминистративныеИнтерфейсыИСЭАР.doc

13.3.6 Документация к WSDL интерфейсу нотификации

Размещается в файле (документе) – АдминистративныеИнтерфейсыИСЭАР.doc

13.4 Приложение. Спецификация административных интерфейсов СЭВ

Приложение описывает административные интерфейсы АИС «СЭВ».

13.4.1 WSDL-определение интерфейса сервиса авторизации

Размещается в файле (документе) – АдминистративныеИнтерфейсыСЭВ.doc

13.4.2 WSDL-определение интерфейса сервиса исполнения

Размещается в файле (документе) – АдминистративныеИнтерфейсыСЭВ.doc

13.4.3 WSDL-определение интерфейса сервиса отладки

Размещается в файле (документе) – АдминистративныеИнтерфейсыСЭВ.doc

13.4.4 WSDL-определение интерфейса сервиса сообщений

Размещается в файле (документе) – АдминистративныеИнтерфейсыСЭВ.doc

13.4.5 WSDL-определение интерфейса репозитария

Размещается в файле (документе) – АдминистративныеИнтерфейсыСЭВ.doc

13.4.6 Документация к WSDL интерфейсу сервиса авторизации

Размещается в файле (документе) – АдминистративныеИнтерфейсыСЭВ.doc

13.4.7 Документация к WSDL интерфейсу сервиса исполнения

Размещается в файле (документе) – АдминистративныеИнтерфейсыСЭВ.doc

13.4.8 Документация к WSDL интерфейсу сервиса отладки

Размещается в файле (документе) – АдминистративныеИнтерфейсыСЭВ.doc

13.4.9 Документация к WSDL интерфейсу сервиса сообщений

Размещается в файле (документе) – АдминистративныеИнтерфейсыСЭВ.doc

13.4.10 Документация к WSDL интерфейсу репозитария

Размещается в файле (документе) – АдминистративныеИнтерфейсыСЭВ.doc