Моделирование бизнес-процессов в ARIS: референтные модели ИТ-компании

Концепция процессного управления является ключевым элементом в методологиях оперативного и стратегического менеджмента, на нее опирается концепция управления качеством TQM. В методологии стратегических карт (Strategy Map) Роберт Каплан и Дэвид Нортон устанавливают связь между целями собственников, которые достигаются путем предоставления ценности для клиентов на основе бизнес-процессов организации, протекающих в заданной культурной, технологической и организационной среде [6]. Таким образом, бизнес-процессы оказываются в центре методологии Strategy Map.

> *Процесс* (от лат. processus - продвижение), 1) последовательная смена состояний стадий развития. 2) Совокупность последовательных действий для достижения какого-либо результата (например, производственный П. - последовательная смена трудовых операций).
>
> *БЭС*

“Бизнес-процесс” -

- структурированный набор действий, охватывающий различные сущности предприятия и подчиненный определенной цели (ISO/CD 15531-1);
- «множество законченных состыкованных работ, которые в совокупности создают продукцию, имеющую потребительскую ценность для клиента» (Дж. Мартин);
- «набор активностей, которые преобразуют несколько видов входных характеристик в выход, имеющий ценность для потребителя» (М. Хаммер, Дж. Чампи) [8];
- специфически упорядоченная во времени и в пространстве совокупность работ, с указанием начала и конца и точным определением входов и выходов» (Т. Давенпорт) [2];
- самый большой элемент, если рассматривать поток работ, который пронизывает всю организацию, начинается у внешних поставщиков и заканчивается у внешних покупателей (DoD, 3);
- «набор из одной или нескольких процедур или действий, которые совместно

Майкл Хаммер и Джеймс Чампи предложили концепцию ре-инжиниринга бизнес-процессов (Business Process Reengineering, BPR), благодаря которой процессное управление было интегрировано в систему управления корпораций [8]. BPR стал способом радикального повышения эффективности бизнеса, снижения издержек и повышения качества услуг. Суть BPR в реинтеграции процессов – корректировка крайности специализации и разделения труда, преодоление отрицательных эффектов подходов Смита-Маркса-Гилбрета.

Определения *бизнес-процессов* являются общими и требуют дополнительного уточнения для сферы разработки ПО. Для этого выделяются области процессов и уровни процессов.

Определение процесса создания автоматизированной системы приводится в ГОСТ 34.601. *Процесс создания автоматизированной системы* представляет собой совокупность упорядоченных во времени, взаимосвязанных, объединенных в стадии и этапы работы, выполнение которых необходимо и достаточно для создания АС (автоматизированной системы), соответствующим заданным требованиям.

Развитие концепции процессного управления обусловило развитие техник графического и аналитического моделирования бизнес-процессов, появление сопутствующего программного обеспечения.

В управлении разработкой ПО концепция управления бизнес-процессами находит свое отражение в моделях зрелости (CMM / CMMI) и стандартах улучшения процессов (SPICE). Управление процессами приводит к улучшению функционирования организации, повышению эффективности отдельных процессов и организации в целом. Управление проектом направлено на выполнение целей проекта с использованием доступных ресурсов и в заданные сроки.

Три уровня усилий по улучшению

- **Непрерывное улучшение процессов** (Continuous Process Improvement, CPI) – устранение вариаций в качестве выпускаемых продуктов и оказываемых услуг.
- **Ре-дизайн бизнес-процессов** (Business Process Redesign) - удаление активностей, не создающих дополнительной стоимости. Сокращение времени выполнения процесса, снижение стоимости.
- **Реинжиниринг бизнес-процессов** (Business process reengineering) – радикальная трансформация процессов с использованием новых технологий для достижения существенного выигрыша в производительности процесса, эффективности и качестве.

Процессы существуют как шаблоны последовательности действий (классы). В ходе выполнения проекта осуществляется реализация процесса - появляется экземпляр процесса – объект (process instance).

Управление процессами включает в себя управление шаблонами процессов (создание моделей процессов) и управление практической реализацией (оперативное управление). Для создания моделей процессов используются формализованные нотации, среди которых выделяется класс графических нотаций.

Три уровня процесса:

- Метапроцесс: - политика, приемы и практика, которые присущи некоторой организации при ведении интенсивного бизнес, связанного с ПО. В центре этого процесса находится экономика организации, долговременная стратегия и возврат инвестиций в ПО.
- Макропроцесс - политика, приемы и практика, которые присущи некоторому проекту по созданию законченного ПО с учетом определенных ограничений по стоимости, срокам и качеству. В центре этого процесса находится создание адекватного варианта метапроцесса для конкретного набора ограничений.
- Микропроцесс - политика, приемы и практика, присущие команде разработчиков некоторого проекта и направленные на получение результатов в процессе создания ПО. Главным для микропроцесса является создание промежуточного продукта адекватного качества с адекватными функциональными возможностями настолько экономично и быстро, насколько это осуществимо на практике

У. Ройс [7].

Под формальным описанием понимается любое структурированное описание процесса, как с использованием средств графического моделирования (IDEF0, EPC, Activity Diagram), так и вербального описания.

Бизнес-процессы пересекают границы функциональных подразделений, выполняются на разных уровнях в организации. Наблюдения показывают, что чем выше уровень управления, тем сложнее идентифицировать процесс, которому принадлежит выполняемая функция. Причиной тому служит совмещение нескольких целей при выполнении одного действия. Все большую часть времени занимает неформализованное общение с сотрудниками и клиентами, обмен информацией относительно условий контрактов, рисков, согласования сроков.

> Три концептуальных вида деятельности в проектах разработки ПО [5]:
>
> - формулирование формальных конструкций (дизайн, проектирование);
> - воплощения в реальном материале (программирование, документирование);
> - диалог с пользователями в реальной жизни (внедрение, поддержка, обучение).
>
> *Ф. Брукс [5]*

Отдельную проблему представляет собой построение процесса, пересекающего иерархические уровни в организации (выходы функций вышестоящего уровня являются входами или механизмом управления функции нижестоящего уровня). Эта проблема осложняется наличием следующих факторов: потеря информации (утаивание, искажение информации, особенно информации об отклонениях и рисках), меньшая структурированность деятельности на более высоких уровнях управления.

Модель процессов представляет собой иерархически упорядоченный список процессов. В классической работе Фредерика Брукса Мифический человеко-месяц выделяются три концептуальных вида деятельности в проектах разработки ПО: формулирование формальных конструкций, воплощения в реальном материале, диалог с пользователями в реальной жизни.

В результате концептуального проектирования создается сущность программы (essence), а результатом воплощения является второстепенная составляющая (accident). Такое разделение на создание сущности и воплощение обусловлено технологией разработки того времени (50-70 годы), отсутствием интерактивных средств разработки, позволяющих объединить проектирование и программирование, а также существенным удельным весом трудозатрат на техническую работу (до 90% времени). Брукс предлагал сфокусироваться на повышении эффективности процессов создания сущности, уделять больше внимания концептуальному замыслу. Сегодня не представляется возможным разделить собственно творческую и техническую работу, кроме того, затраты на техническую работу постоянно сокращаются.

Для повышения производительности труда разработчиков Ф. Брукс считал необходимым выделить затраты, необходимые для изготовления концептуальных конструкций и для точного и упорядоченного их представления.

Для практического использования процессного подхода и понимания взаимосвязи между концепциями управления и процессами в организации имеет смысл выделить следующие три группы процессов:

- *Процессы управления проектами* — планирование, организация и контроль проекта;
- *Процессы разработки продукт* — проектирование, разработка, тестирование и внедрение ПО, разработка документации;
- *Организационные процессы* – направленные на улучшение работы всей компании, процессы обучения, процессы изменений.

Процессы управления проектами являются типовыми, их регламентирует методология управления проектами. Процессы разработки продукта связаны с вы-

бранной моделью ЖЦ ПО, которая определяется выбором методологии управления процессом разработки ПО. Некоторые проектные методологии содержат детальное описание процессов.

Все процессы в ИТ-компании делаться на две группы: связанные и не связанные непосредственно с проектами разработки ПО. При составлении плана проекта учитываются только те функции (работы), которые непосредственно связаны с проектированием и разработкой. Связь с остальными функциями осуществляется через доступность ресурсов.

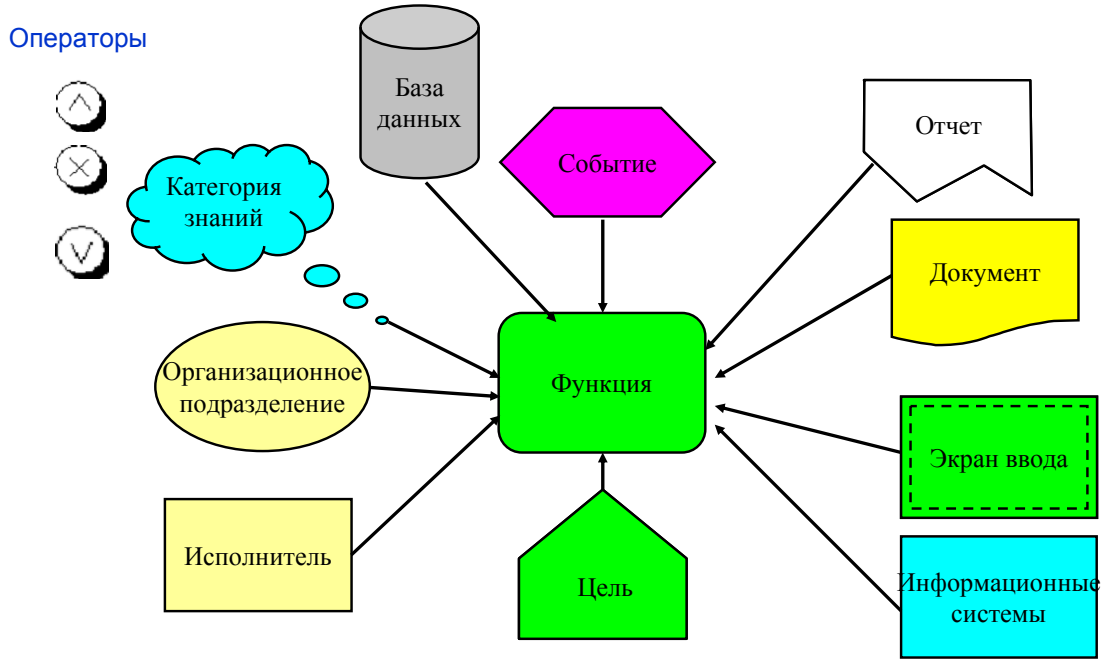
Отсутствие формализованного описания процесса, отсутствие повторяемости процессов представляются значимой проблемой, с которой сталкиваются организации-разработчики ПО.

Проблема формализации бизнес-процессов актуальна для всех компаний и организаций. Внедрение информационной системы и (или) сертификация системы менеджмента качества требуют формализации бизнес-процессов. Формальное описание процессов необходимо для обеспечения повторяемости процесса, однозначного понимания процессов всеми сотрудниками, возможности измерения характеристик процессов.

Для компаний-разработчиков ПО проблема формализации бизнес-процессов (процессов разработки) отличается от той проблемы, с которой сталкиваются производственные компании, поскольку повторяемость процессов в производственных компаниях, как правило, наблюдается и в том случае, когда отсутствует их формальное описание. В разработке ПО *отсутствие формального описание в абсолютном большинстве случаев свидетельствует об отсутствие их повторяемости* (незрелости процессов, в терминологии CMM).

Однозначное выделение (определение) процессов в функционирующей организации представляет техническую сложность, обусловленную наличием связей между процессами, отсутствием регулярной повторяемости процесса, существованием «висящих» функций – функций, которые не принадлежат ни одному процессу или функций, принадлежащих нескольким процессам одновременно.

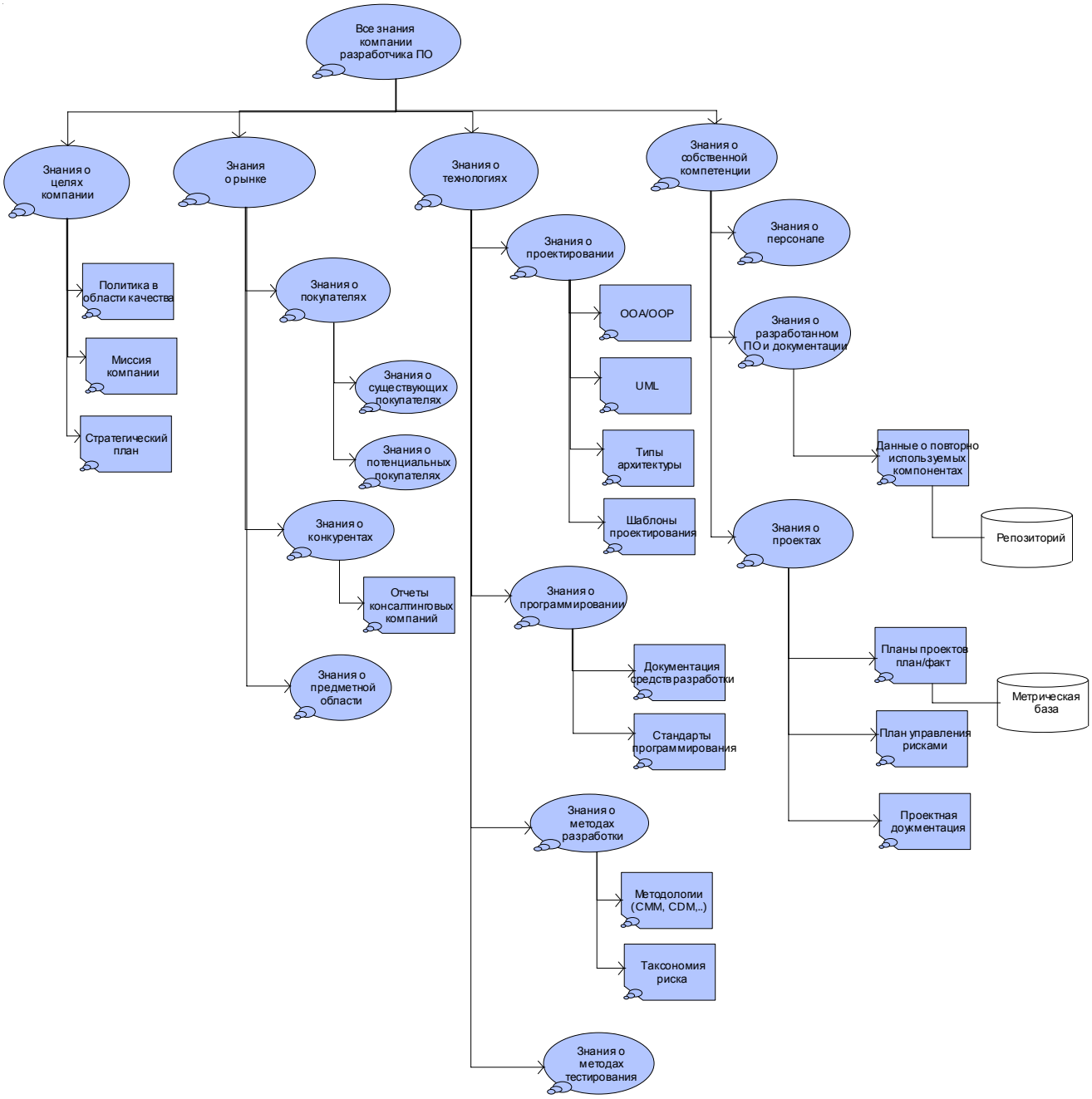
Для моделирования процессов используется нотация ***ARIS eEPC.***



Нотация ARIS eEPC.

Модель структуры знаний и карта знаний

Модель стриктуры знаний компании является наиболее общей, в сравнении с моделями процессов, моделью организации. На ней могут быть представлены как неструктурированные и недокументированные знания (как правило, наиболее общие), так и знания, содержащиеся в документах, базах данных и хранилищах данных. Модель структуры знаний позволяет выявить знания, существенные для работы компании, определить, где содержаться эти знания. Не смотря на то, что модели знаний на прямую не связаны с процессом, их использование при формализации и улучшении процесса оправдано, поскольку в процессах формируются и используются знания, содержащиеся в документах, база данных, и недокументированные знания.

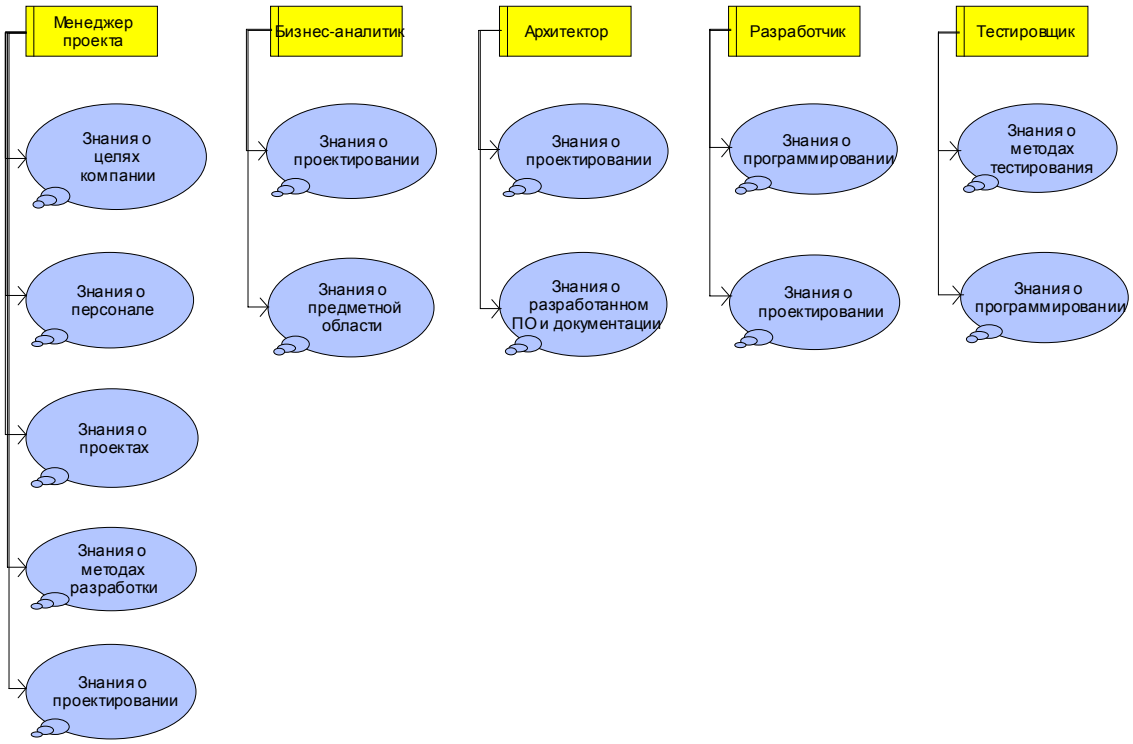


Модель структуры знаний

Процесс обучения должен быть направлен на расширение знаний, а одна из задач процесса создания инфраструктуры состоит в обеспечение доступа к знаниям в любой форме. Таким образом, модели знаний позволяют выделить некоторые ар-

тефакты (документы), которые будут использоваться при проектировании процессов, а так же определяют требования к процессам.

Карта знаний показывает роли, которые играют участники проекта, и знания, которыми они обладают (или которые им необходимы для выполнения работы). Следует отметить, что построение моделей является итерационным процессом, и в ходе моделирования процессов разработки уточняется и дополняется структура знаний и карта знаний.

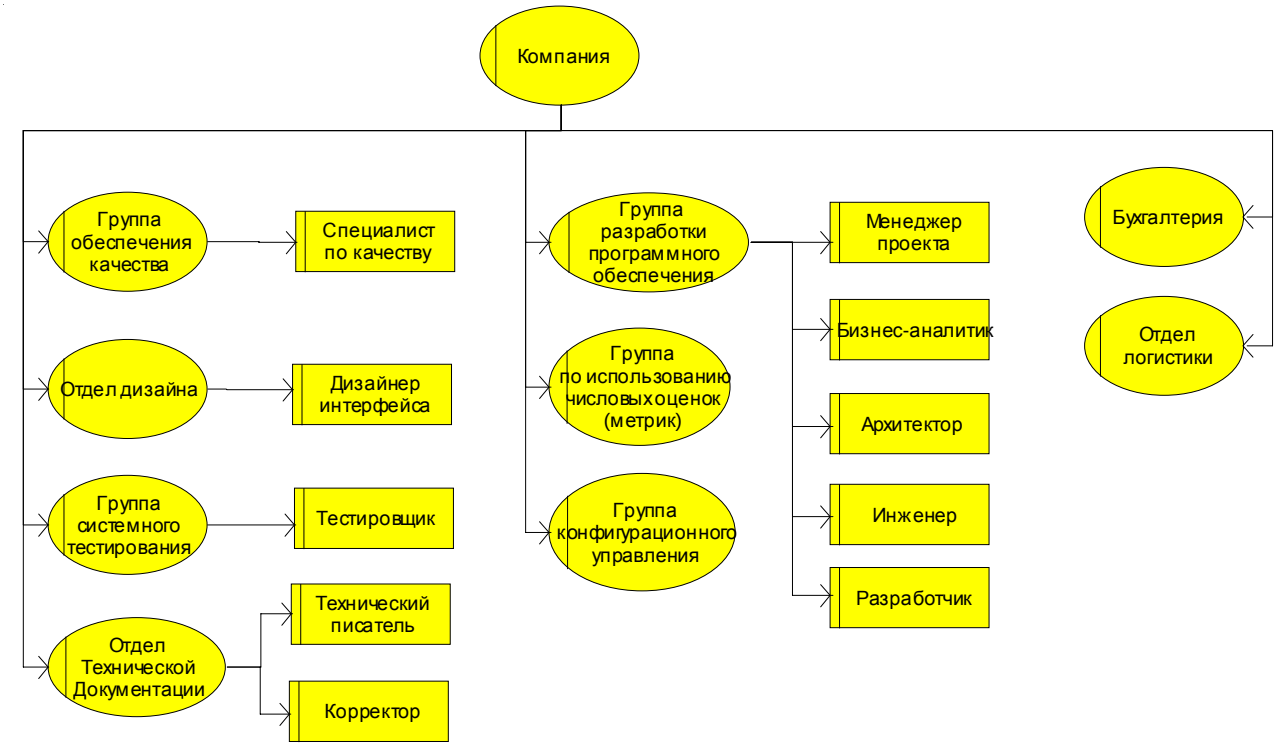


Карта знаний

Большинство компаний-разработчиков ПО имеют матричную или проектную организационную структуру. Предлагаемая структура организации содержит как проектные группы, так и функциональные отделы. При этом участники проектных групп не входят в существующие отделы, а сотрудники отделов участвуют в выполнении проектов.

Модель организационной структуры используется для построения модели процессов разработки, поскольку она выделяет отделы и исполнителей, ответственных за выполнение функций.

Модель организационной структуры



Модели процессов разработки жизненного цикла

Представленные в работе референтные модели могут быть детализированы организацией до уровня отдельных бизнес-процедур, должностных инструкций. Представленные модели процессов реализуются на тех фазах, на которых они являются основными (если используется MSF). В общем случае, модели процессов будут отличаться для различных этапов ЖЦ ПО

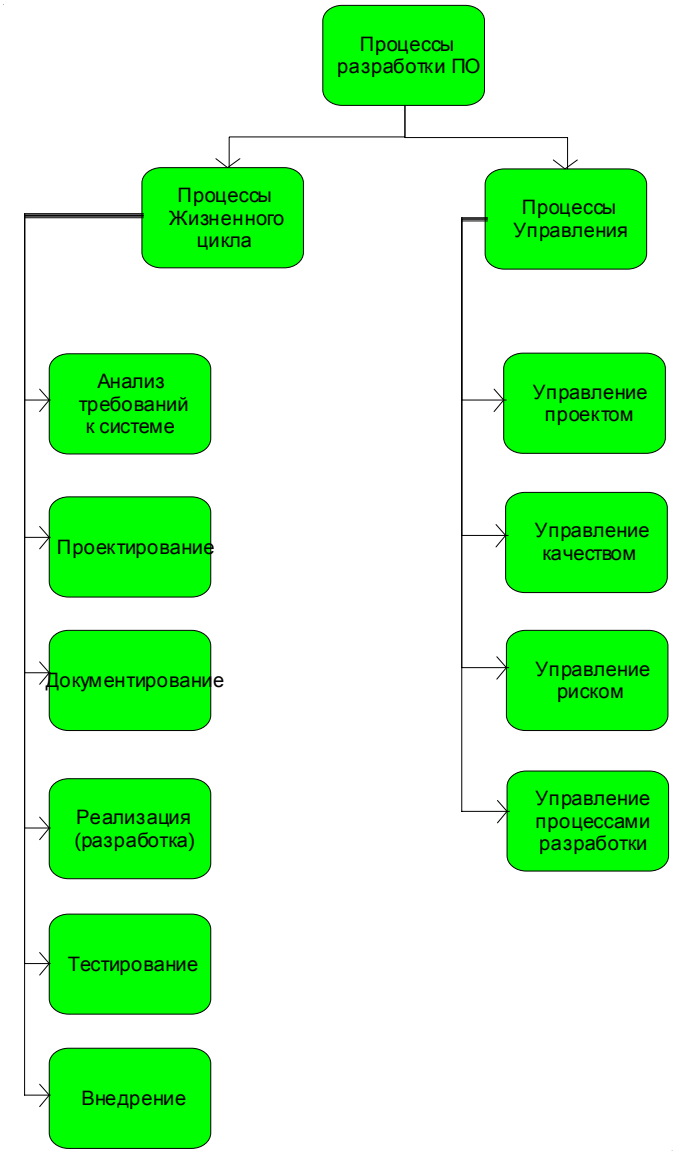
Дерево функций (структура процессов)

Предлагаемая модель процессов отличается от всех моделей, рассмотренных в работе. В отличие от ISO 12207 предлагается не три, а две группы процессов. На наш взгляд процессы обучения, усовершенствования и создания инфраструктуры являются достаточно общими, они не специфичны для разработки ПО, поэтому мы не включили их в набор референтных моделей.

Не смотря на то, что предлагаемые модели процессов в первую очередь предназначена для разработчиков заказных информационных систем для бизнеса, процесс конвертирования данных не выделен в отдельный процесс, как предлагается в CDM. На наш взгляд, конвертирование данных является частью процессов анализа требования, проектирования, разработки и тестирования. Можно рассматривать перенос данных как одно из функциональных требований.

Поддержка и сопровождение не является в полной мере процессом разработки ПО, включает в себя консультации пользователей, отслеживание изменений в требованиях государственных организациях. Безусловно, поддержка пользователей (текущее обслуживание) необходимо в конечном итоге рассматривать как единый процесс, так как для него существует общие входы (договор на поддержку), выходы (сервис для клиентов). В рамках этого процесса следует выделять подпроцессы - исправление ошибок, доработки и консультации.

Кроме представленных на карте процессов можно так же выделить процесс управления изменениями.



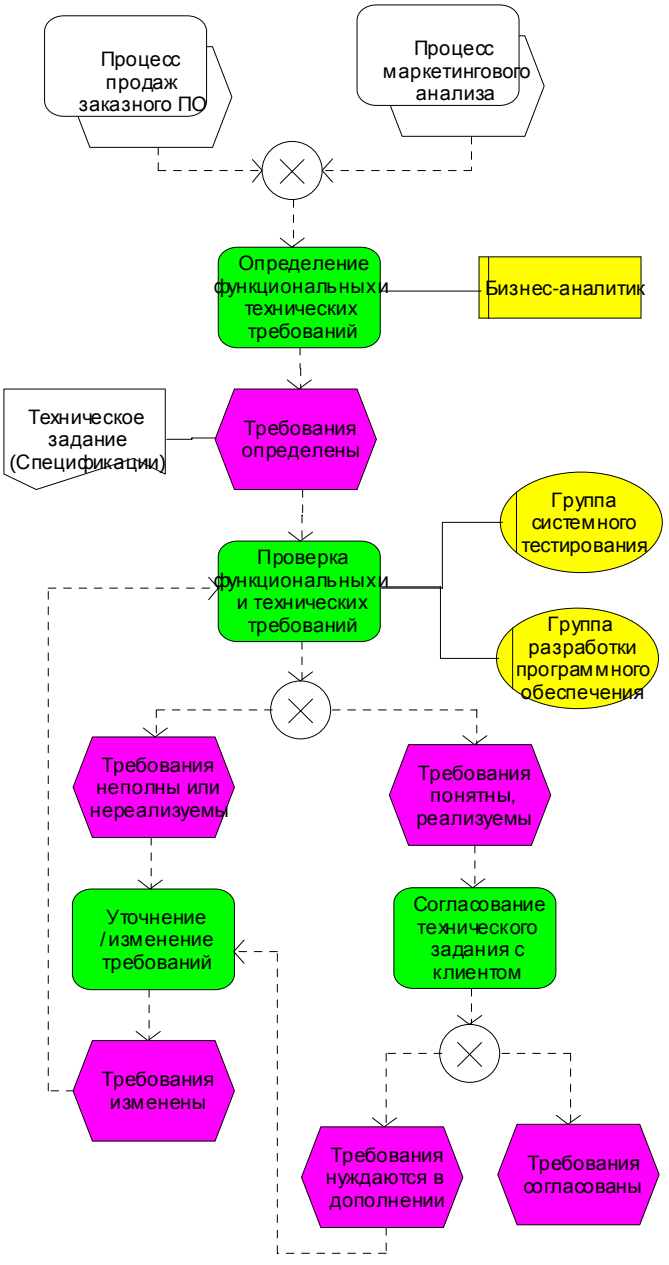
Дерево функций (структура процессов)

Процесс анализа

Процесс анализа включает в себя определение требований заказчика (для заказного ПО) или определите требований рынка. Процесс анализа совпадает со стадией концептуального дизайна процесса проектирования MSF. Целью процесса является учет требований пользователей и бизнеса. В результате процесса получается описание задачи и ее решение в терминах сценариев. Кроме функциональных требований определяются требования к переносу данных, надежности, техническому обеспечению. Полный набор требований содержится в стандарте на техническое задание. Определяются контрактные условия. Входом процесса является договор на разработку с заказчиком или заявка на разработку от отдела маркетинга, которые являются выходами процесса продаж или маркетингового анализа.

В ходе процесса требования формализуются и согласовываются с заказчиком (на предмет полноты) и с группами разработки и тестирования (на предмет понятности, исполнимости и согласованности). Здесь могут быть использованы метрики качества требований. Все требования, для которых были обнаружены проблемы, просматриваются бизнес-аналитиком (или группой, ответственной за анализ), в документацию вносятся изменения.

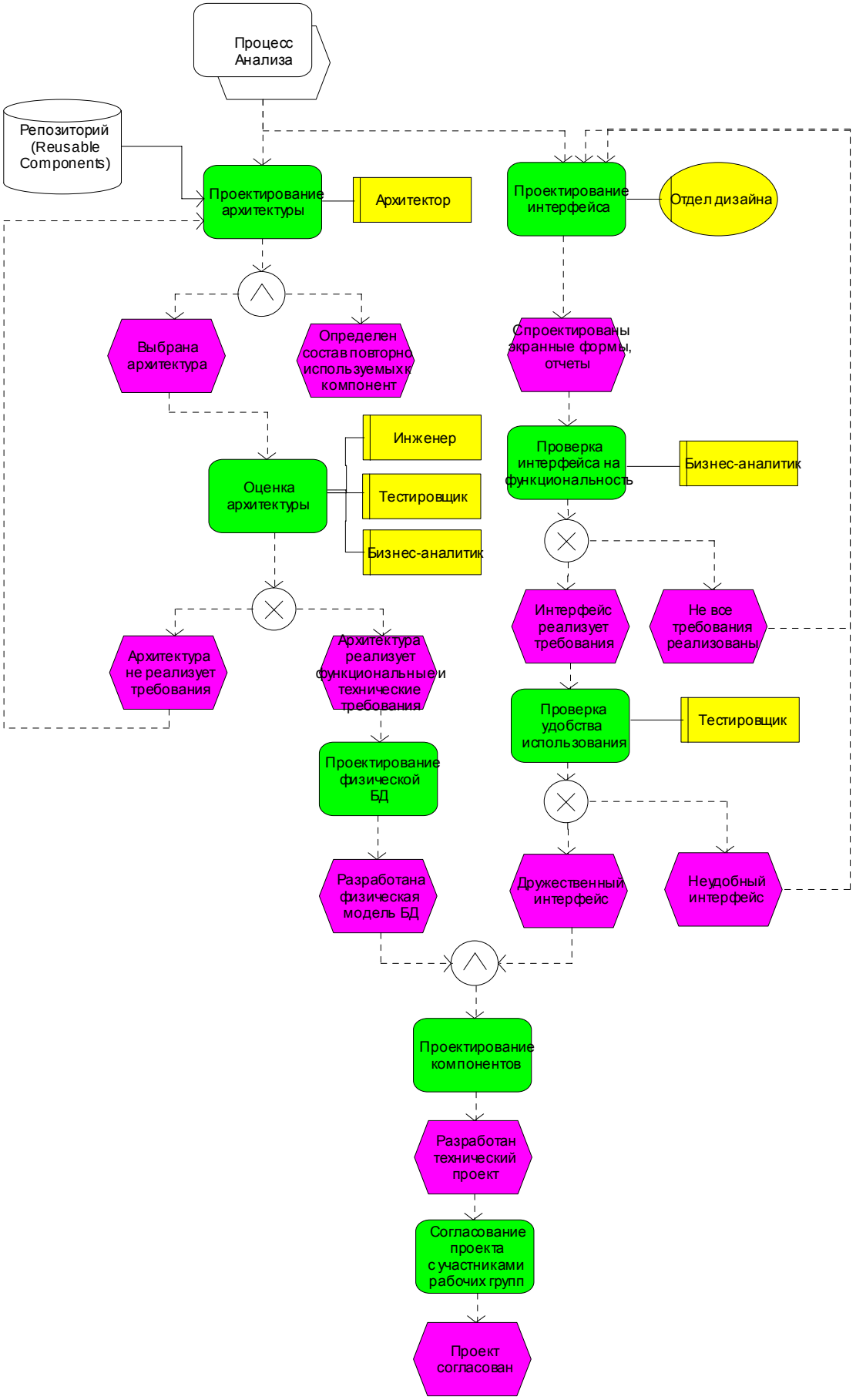
Предлагаемая модель требует согласования функциональных и технических требований с группами разработки, системного тестирования для оценки их понятности, выполнимости, согласованности.



Процесс анализа

Процесс проектирования

Процесс проектирования необходим для получения согласованного плана разработки. С одной стороны технический проект представляет собой техническое решение, а с другой – является продолжением планирования.



Процесс проектирования

Процесс проектирования включает в себя логическое проектирование, состоящее в учете требований проектной группы и представленное как набор серви-

сов (абстрактная модель решения) и физическое проектирование, в результате которого определяется технология разработки (спецификации интерфейса, физическая модель БД).

В зависимости от выбранной модели ЖЦ процесс проектирования может быть инициирован завершением процесса анализа, предоставлением согласованных требований к ПО (для каскадной модели) или начаться одновременно с процессом анализа (для MSF).

Предлагаемая модель процесса проектирования предполагает выделение дизайна интерфейса в отдельный процесс, который предшествует разработке компонент.

Процесс разработки

Процесс разработки имеет на входе технический проект, а на выходе – готовую к установке версию ПО (на промежуточных циклах собирается версия для системного тестирования, на последней итерации – дистрибутив для установки у заказчика). Процесс разработки включает в себя создание приложения, реализующего интерфейс и объекты бизнес-логики, создание БД, хранимых процедур и триггеров.

Можно выделить несколько различных уровней детализации процесса разработки (как и любого другого процесса). На самом высоком уровне определяется логика процесса, детализированная до функций, выполняемых группой разработки. На следующем уровне детализируются функции, выполняемые отдельным разработчиком в течение одного цикла (от получения требований до сборки версии для системного тестирования). Дальнейшая детализация приводит к рассмотрению функции создания (модификации) программного объекта (класса, хранимой процедуры) как отдельного процесса. При построении этого процесса использовался PSP. Выделяются такие функции как блокировка (разблокировка) объекта в средстве управления версиями, просмотр собственного кода, тестирование компонент с использованием списка ошибок. Конечный уровень детализации процесса определяется желаемым уровнем регламентации работы, детализацией инструкций.

Существуют следующие аспекты, которые определяют требования к эффективному процессу разработки:

Необходимо использовать средства управления версиями вне зависимости от того, сколько человек работает над проектом. Средства управления версиями отличаются своей сложностью (поддерживаемыми функциями). К функциями управления версиями, выполняемым в процессе разработки, относятся: помещение приложения в средство управления версиями, создание локальной копии приложения, блокировка / разблокировка объекта для внесения в него изменений, помещение измененной версии в средство управления версиями. Средство управления версиями позволяет исключить одновременное изменение одного объекта двумя разработчиками, позволяет проводить тестирование сделанных изменений в локальной версии, обеспечивает хранение всех версий объекта. Наиболее продвинутые средства управления версиями поддерживают несколько сборок (baseline), позволяют иметь несколько версий приложения одновременно (например, версию, поставляемую пользователю, версию для системного тестирования и новую версию). Visual Source Safe - наиболее часто используемое небольшими компаниями-разработчиками средство управления версиями, не поддерживает одновременную сборку нескольких версий в явном виде. Следует строить процесс разработки в части управления версиями с учетом возможностей используемого средства управления версиями. Отдельную проблему представляет собой управление версиями баз данных (как для самих данных, так и для хранимых процедур и триггеров). Эта

проблема особенно важна для тех СУБД, где данные и процедуры представляют единое целое – хранятся в одном файле (например, MS SQL Server).

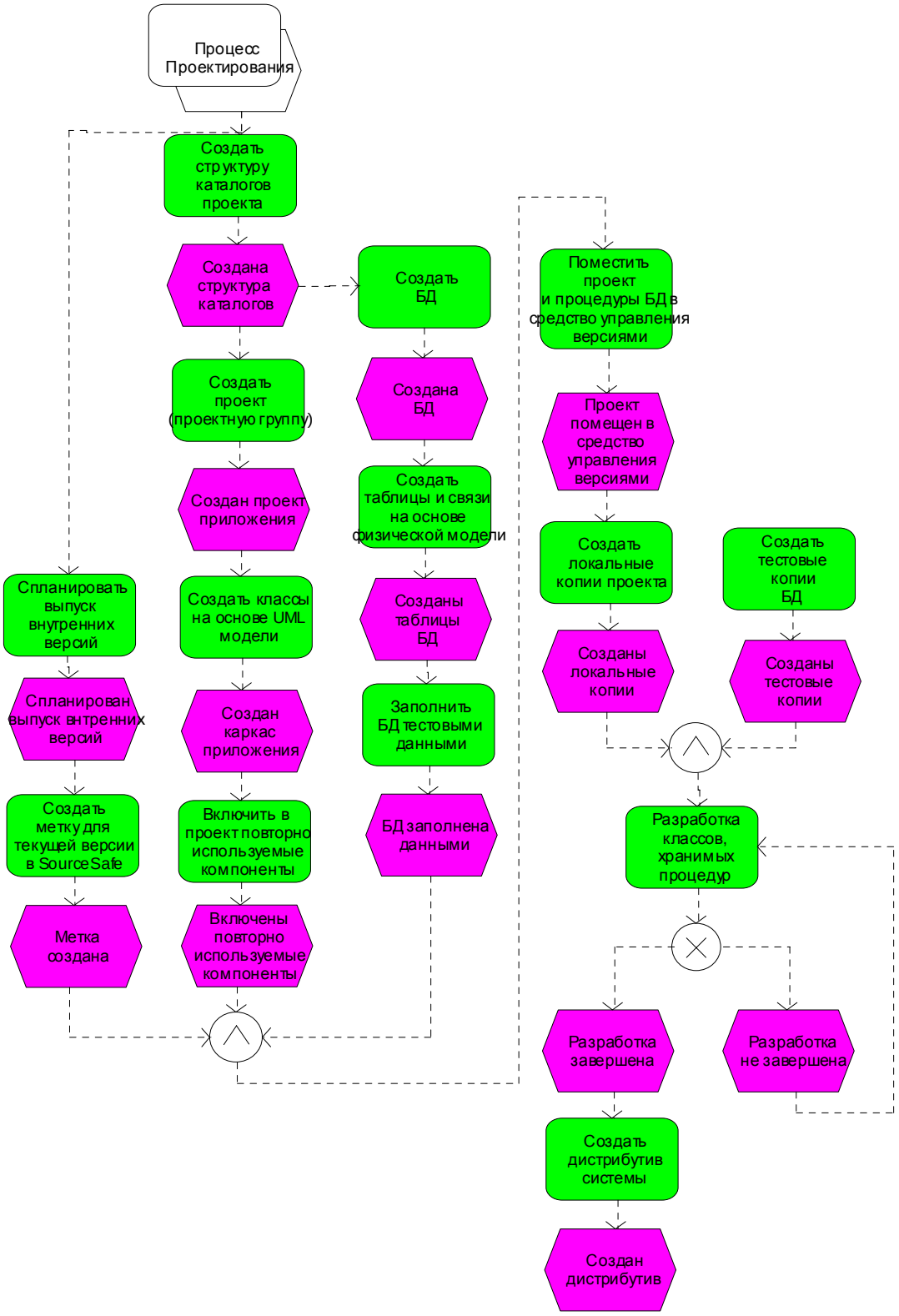
В ходе тестирования собственного кода программист может изменить данные таким образом, что они будут непригодны для дальнейшего использования (например, будет разрушена целостность данных или удалены справочники). Следовательно, необходимо предусмотреть функцию восстановления данных, или использовать копии базы данных в тестировании. Во втором случае так же необходимо поддерживать актуальность тестовых БД.

Необходимо обеспечить интеграцию процесса разработки с другими процессами (тестирования, планирования, документирования и т.д.) Эта интеграция обеспечивается за счет общих событий (например, сборка дистрибутива – процесс разработки инициирует тестирование дистрибутива – процесс тестирования), за счет общих информационных ресурсов (например, новые типы ошибок, выявленные разработчиками, попадают в БД ошибок и используются в процессе тестирования) или за счет общих функций, когда результаты функции приводят к событиям, принадлежащим разным процессам. Этот аспект имеет наибольшее значение для процесса разработки, поскольку он находится в центре совокупности процессов компании.

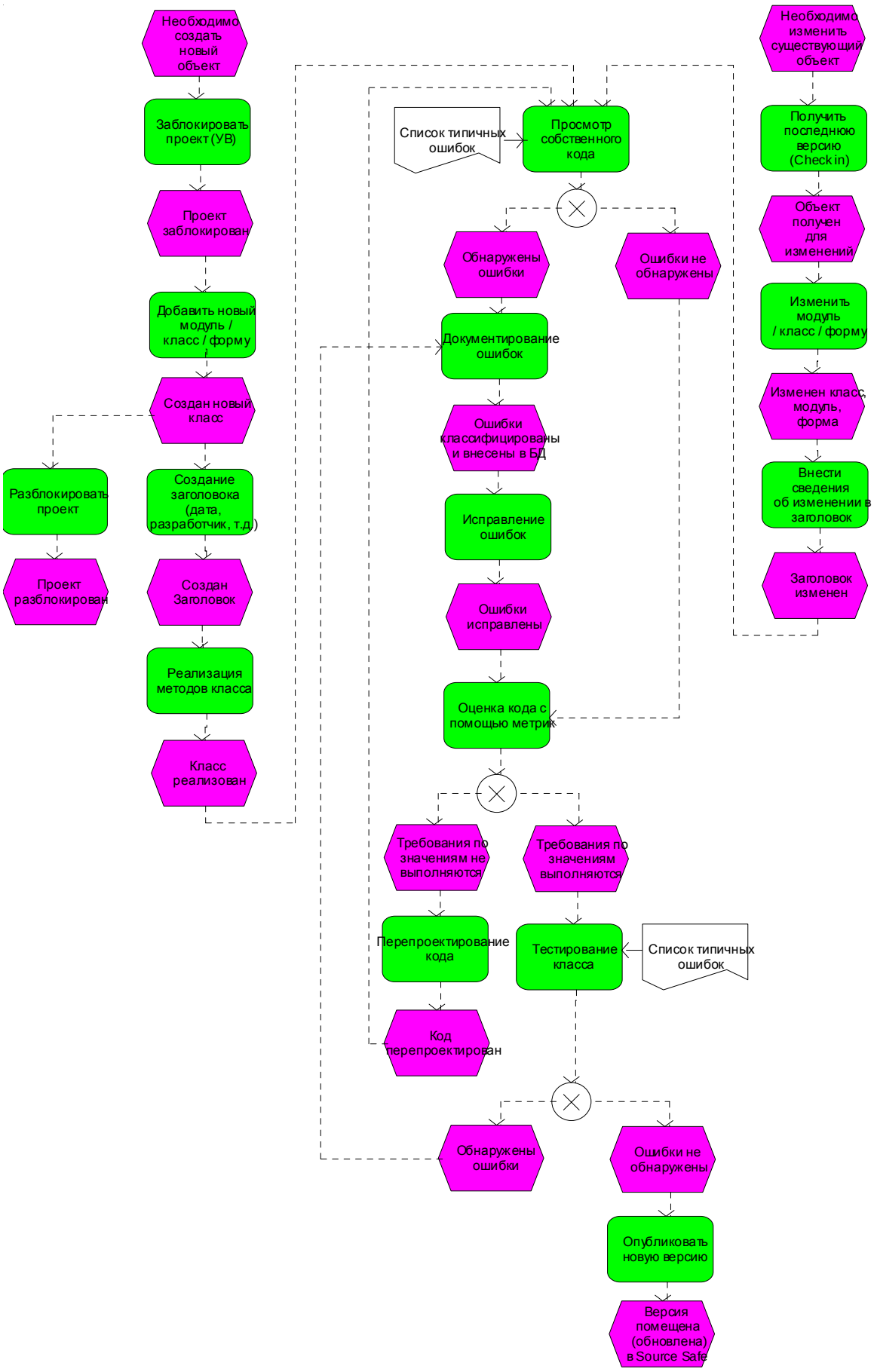
Необходимо обеспечить несколько уровней планирования работы. На самом высоком уровне используется план, получаемый в результате процесса управления проектами: План ЖЦ ПО и План разрабатываемой функциональности. На основе этого плана создается план разработки компонент. Этот план создается совместно группой разработки, группой тестирования и менеджером проекта. На основе плана разработки компонент группа разработки формирует внутренние планы – планы выпуска внутренних версий (не обязательно формальные) в которых определяется, что будет разработано в течение недели. Эти планы составляются самими разработчиками и являются их обязательствами.

Предлагаемый процесс разработки учитывает специфику разработки информационных систем для бизнеса. Заполнение БД данными вынесено в отдельный подпроцесс. Описан порядок создания структуры каталогов, создание шаблона приложения из модели, разработанной в процессе проектирования. Установлен порядок включения повторно-используемых компонент.

Предлагаемый собственный процесс разработки имеет следующие преимущества: контролируется блокировка / разблокировка модуля проекта, просмотр кода предшествует его тестированию, сведения об ошибках документируются, при просмотре и тестировании используется список типовых ошибок, оценка кода с помощью метрик осуществляется так же до тестирования. На наш взгляд следует просматривать и оценивать код до тестирования, поскольку тестирование занимает значительную часть времени разработчиков, и если после завершения тестирования код не будет удовлетворять требованиям к понятности, сложности, то его изменение приведет к необходимости повторения тестирования, и, следовательно, к потере результатов предыдущего тестирования. На практике это приводит к тому, что если просмотр кода и использование метрик качества и показывают необходимость перепроектирования кода, этого не происходит, поскольку не тестирование уже потрачено достаточно много времени.



Процесс разработки (общий)



Собственный процесс разработки

Процесс тестирования

Тестирование частично включается в процесс разработки. Процесс тестирования выделяется в отдельный процесс, поскольку он имеет свои задачи и выполняется отдельной группой (группой тестирования). Независимость группы тестирования от группы разработки и менеджера проекта является одним из условий обеспечения качества. Тестированию подлежат все разрабатываемые продукты: документация, программное обеспечение. Тестирование ПО включает в себя как тестирование качественных характеристик ПО, так и тестирование дистрибутива системы на различных платформах (на которых будет использоваться система), тестировании документации.

Процесс тестирования начинается раньше, чем процесс проектирования. После того, как в процессе анализа создаются функциональные и технические требования, и еще до того, как эти требования будут полностью согласованы с заказчиком и группой разработки, создается первая версия плана тестирования. Это необходимо для того, что бы учесть затраты времени на тестирование при составлении плана ЖЦ ПО, согласования даты выпуска с заказчиком. Кроме того, в некоторых случаях может потребоваться приобретение дополнительных инструментальных средств тестирования или разработка специализированного ПО. Группа тестирования может заказывать создание ПО для тестирования группе разработки или осуществлять разработку самостоятельно.

В тестировании используются метрики для предсказания времени тестирования, ожидаемой продолжительности тестирования, оценки рисков (например, метрики полноты спецификаций).

Оптимизация тестирования и отладки приложений представляется одной из наиболее актуальных задач. В отличие от этапа проектирования и разработки, длительность которых определяется требованиями к конечному продукту и производительностью, длительность отладки может быть установлена произвольно. То есть, может быть задано время отладки или условие (условия), при котором отладка заканчивается. При этом существующие модели строятся на основе предположения, что между временем отладки и количеством необнаруженных ошибок существует обратная зависимость.

Необходимо различать *первичные* и *вторичные* проявления ошибки. Первичные ошибки обнаруживаются в тексте программ и подлежат корректировке. Вторичные ошибки представляют собой искажение выходных результатов исполнения программы, именно с вторичными ошибками сталкивается конечный пользователь на этапе эксплуатации программы. Следовательно, именно прогнозируемое количество вторичных ошибок должно быть критерием при принятии решения о продолжении или прекращении отладки.

При этом ожидаемое количество вторичных ошибок определяется ожидаемым количеством первичных ошибок и структурой приложения. Например, если в процедуре, используемой для формирования десяти различных отчетов содержится одна ошибка, то пользователь столкнется с десятью ошибками.

Для вторичных ошибок значимыми являются:

- место возникновения (бизнес-функция, отчетность, и т.д.)
- влияние на работу пользователя (отказ, снижение работоспособности, искажение данных);
- частота проявления ошибки (связана с необходимой частотой выполнения операции, которая приводит к ее проявлению).

Таким образом, следует не только учитывать количество обнаруженных первичных ошибок, но прогнозировать вторичные ошибки, уделяя особое внимание

наиболее часто используемым функциям, которые необходимы для бизнес-процесса клиента.

Кроме того, следует различать ошибки и недоработки. Недоработки не приводят к существенному ухудшению работы программы, и, в худшем случае, снижают эстетическое восприятие. Примером недоработки является отсутствие выравнивания полей формы, недостаточный размер поля в отчете для символьной строки. Время, необходимое для обнаружения и исправления всех недоработок, может быть сопоставимо со временем разработки приложения.

Процесс управления рисками тесно интегрирован с процессом тестирования. Именно в ходе тестирования требований, проектных решений и реализации оцениваются возможные риски и вырабатываются меры по их предотвращению.

Существуют два уровня детализации процесса тестирования: формализация процесса в рамках жизненного цикла ПО, при этом делается акцент на тестировании различных продуктов, документации, дизайна интерфейса, проекта, исполнимого кода и формализации отдельной операции тестирования, акцент делается на использовании метрик для прогнозирования значения и фиксации полученных результатов для использования в других проектах и улучшения процессов разработки.

Большую долю времени в процессе тестирования занимает подготовка сценариев и наборов тестовых данных. Следует отметить, что тестовые данные необходимы также группе разработки для проведения внутреннего тестирования. Поэтому необходимо синхронизировать процесс разработки и тестирования – тестовые данные должны быть подготовлены до начала разработки компонент. Тестовые данные могут подготавливаться как группой разработки, так и группой тестирования, в зависимости от степени загрузки сотрудников, учитывая многопроектный характер работ.

Наибольшую сложность в моделировании процессе тестирования представляет его циклический характер, обусловленный тем, что при исправлении ошибок, найденных в результате тестирования, могут быть внесены новые ошибки, которые так же необходимо обнаружить, и исправить. В крайнем случае, процесс может быть несходящимся, это произойдет в том случае, если при исправлении ошибок будет вноситься больше ошибок, чем было исправлено. Очевидно, что моделируя отдельно процесс разработки и тестирования, достаточно сложно отобразить существующее между ними взаимодействие.

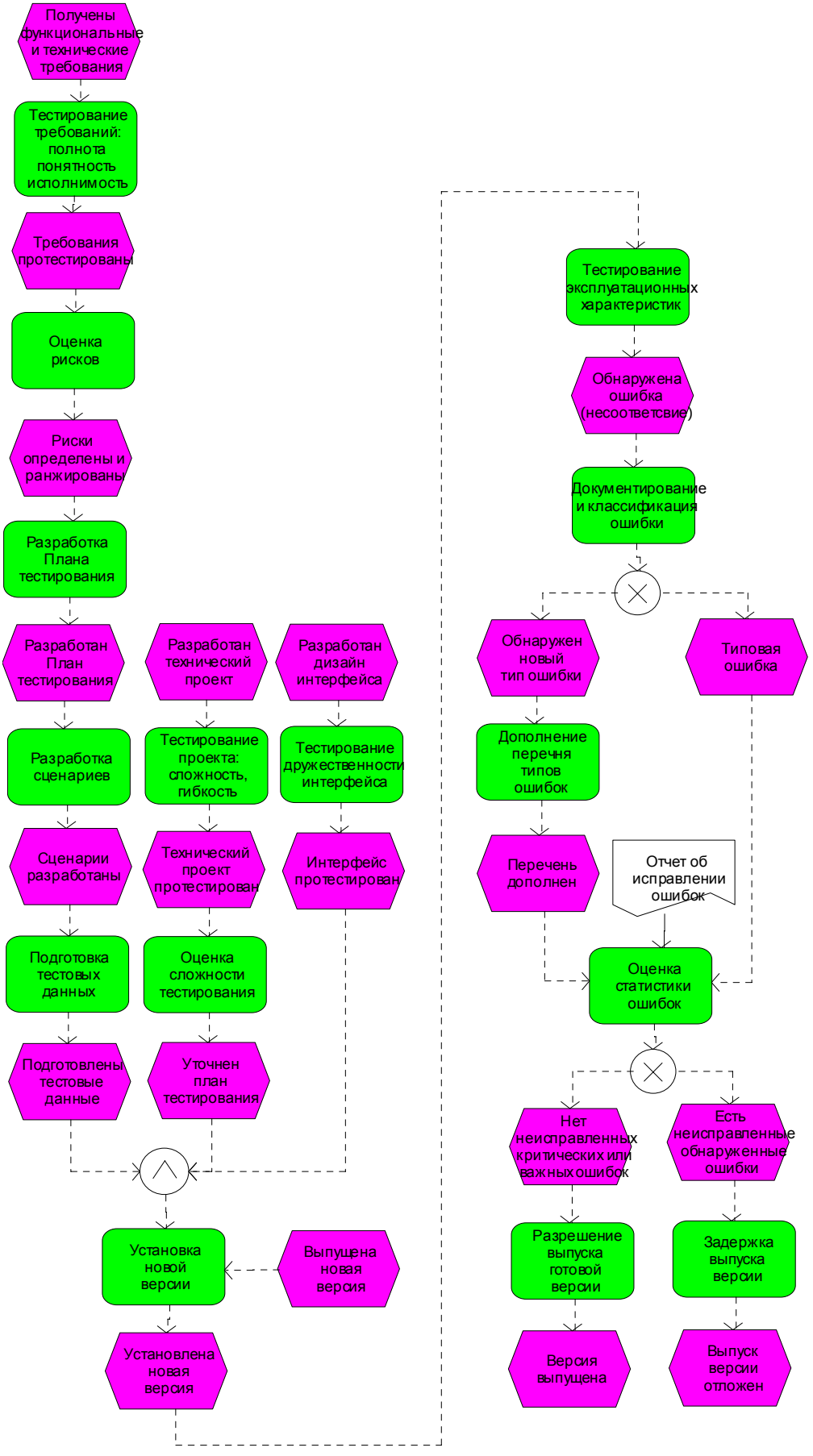
Можно выделить несколько этапов процесса тестирования, подготовка тестирования: составление плана тестирования, разработка сценариев, подготовка тестовых данных, тестирование требований, тестирование проекта, тестирование ПО и документации: выполнение сценариев, тестирование дистрибутива. В каждом случае при обнаружении ошибок в документации или ПО, а также при обнаружении несоответствия характеристик заданным требованиям группа тестирования документирует обнаруженные ошибки и несоответствия, присваивает им уровень важности. Кроме того, обновляется БД типов ошибок, используемая при составлении сценариев тестирования.

Исправление ошибок может занять много времени, которое может быть неучтено в плане проекта. Использование методов прогнозирования времени тестирования для корректировки планов является перспективным направлением улучшения процессов.

Кроме того, можно выделять в самостоятельные процессы процесс тестирования кода, документации, дистрибутива (полноты), производительности. Предлагаемая модель процесса не раскрывает циклический характер процесса тестирования, в модели сознательно опущено тестирование документации. Модель охватывает полностью ЖЦ ПО: от подготовки требований, до выпуска готовой версии. Предусматривается оценка рисков на основе анализа требований, разработка пла-

на тестирования. Следует отметить, что план тестирования может повлиять на план разработки (например, в первую очередь будут разработаны те компоненты, тестирование которые займет больше времени).

Основная итерация в рамках процесса тестирования осуществляется в соответствии с планом выпуска внутренних версий. В зависимости от масштаба приложения такая версия может разрабатываться с периодичностью от одного дня до одной недели. Внутренняя версия собрана, когда все входящие в нее компоненты протестированы разработчиком.



Процесс тестирования

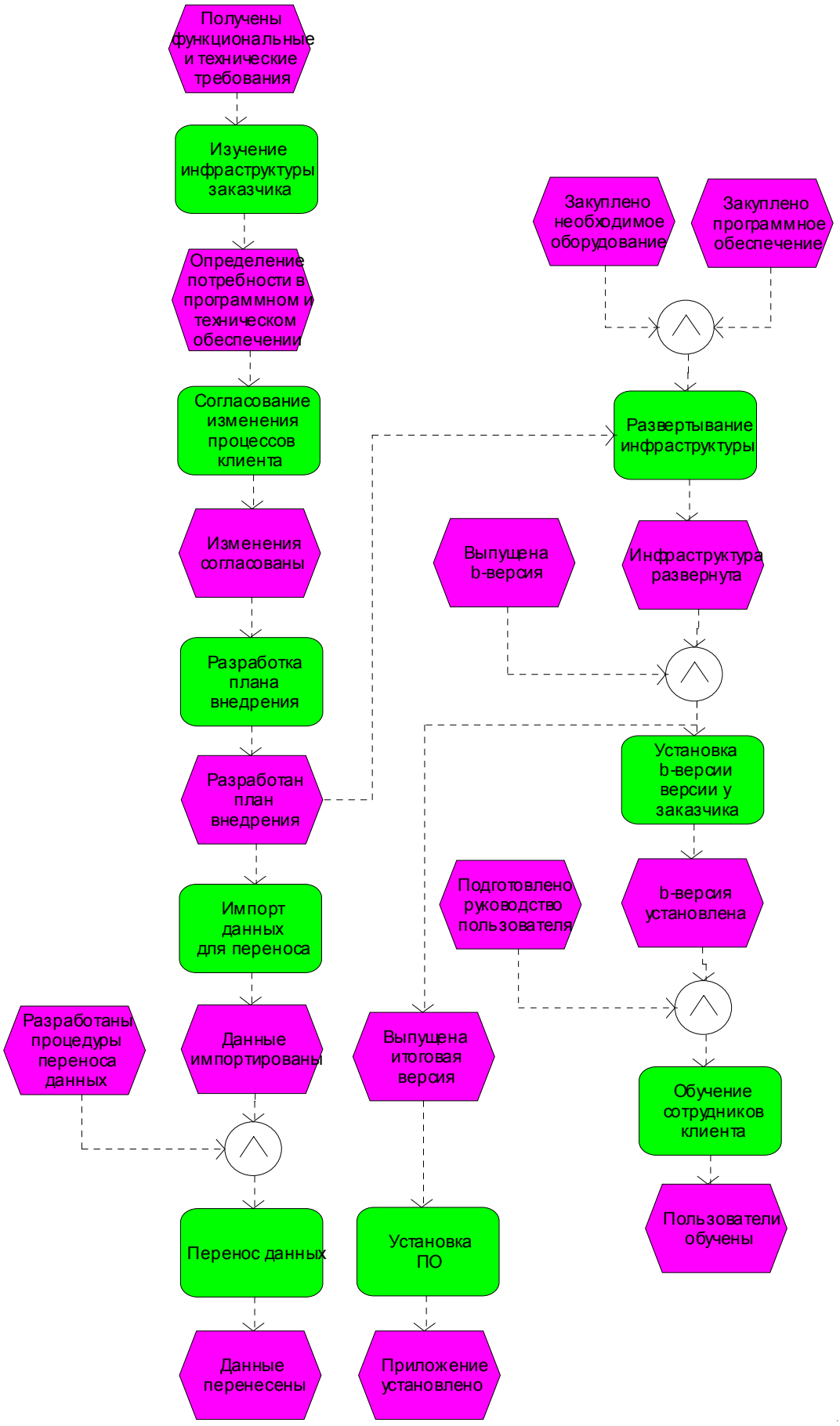
Процесс внедрения

Процесс внедрения начинается задолго до завершения тестирования. Внедрение требует создание плана развертывания. Внедрение, на наш взгляд, представляет собой один из наиболее рискованных процессов, поскольку здесь результат во многом зависит от клиента, его способности предоставить необходимые технические ресурсы, специалистов. Требуется согласование этапов внедрения не только с процессом разработки и тестирования, но и с событиями, происходящими у заказчика.

Необходимо управлять процессом внедрения, то есть процесс управления проектом должен быть тесно интегрирован с процессом внедрения. Следует прогнозировать риски внедрения на ранних этапах разработки, строить сам план ЖЦ ПО с учетом этих рисков, учитывая возможность поэтапного перехода на новую систему, необходимость обучения, требования к переносу данных, а так же необходимость изменения бизнес-процессов заказчика.

Необходимо проанализировать, каким образом изменятся процессы заказчика после внедрения разрабатываемого ПО, согласовать эти изменения до эксплуатации, убедиться, что заказчиком подготовлены новые версии инструкций и процедур.

Представленная модель процесса внедрения предусматривает установку бета-версии ПО до начала обучения пользователей, что бы обучение происходило на системе, максимально похожей на ту, которой будет пользоваться заказчик. Поэтому инфраструктура должна быть развернута как можно раньше. Кроме того, пользовательская документация (руководство пользователя, файл контекстной помощи) должна быть разработана до начала обучения. Процесс внедрения имеет несколько выходов: обученных пользователей, установленную систему и перенесенные данные.



Процесс внедрения

Процесс документирования

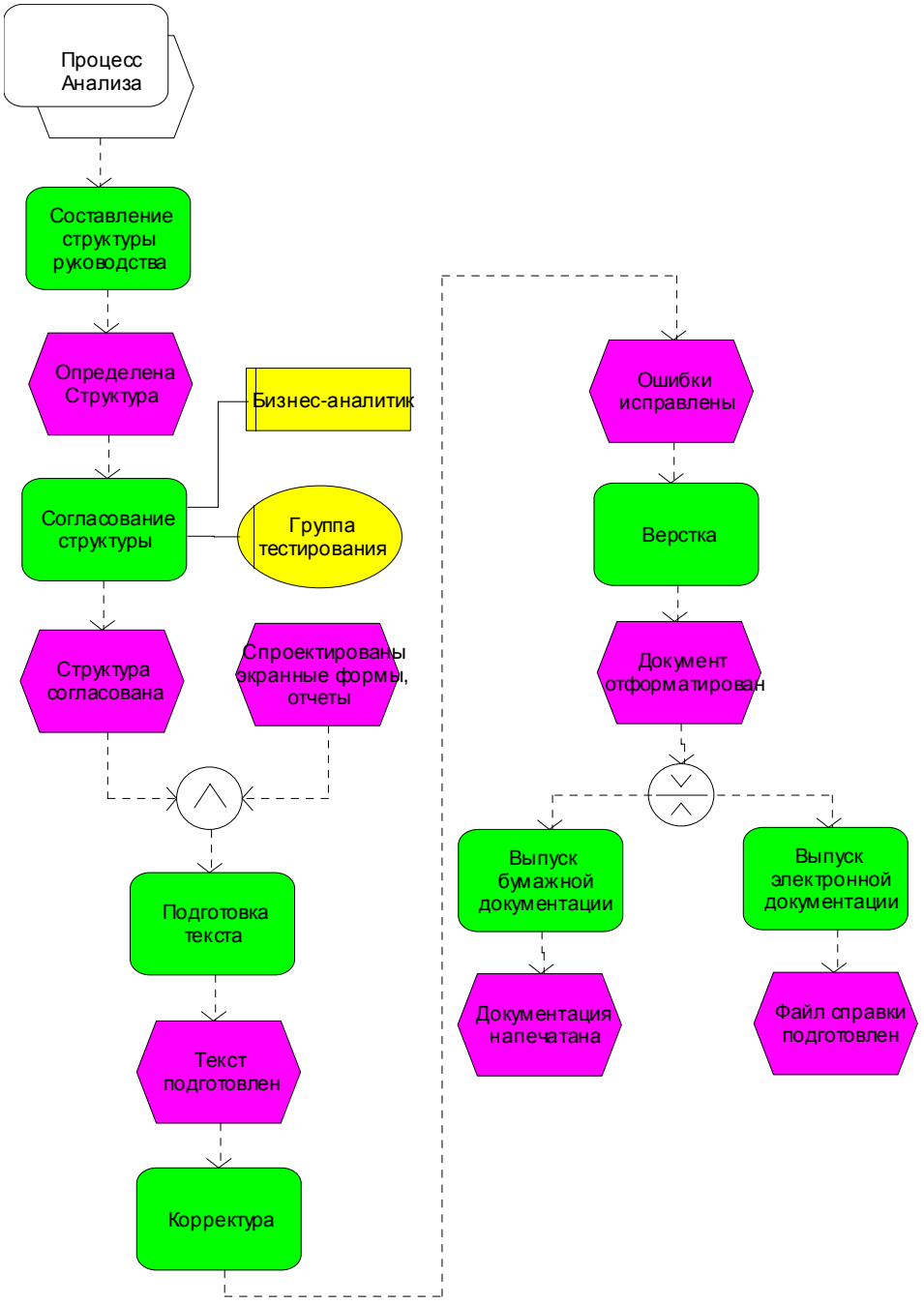
Процесс документирования вынесен в отдельный процесс, поскольку в средних и крупных организациях документированием занимается выделенное подразделение. Составление документации требует специализированных навыков, инструментов и нуждается в организованном процессе. Документирование должно быть синхронизировано с процессом проектирования. Требования к разрабатываемой документации содержаться в техническом задании. Документация может подготавливаться для распространения в бумажном и (или) электронном виде. Специфическим видом документации является электронная справка.

Следует разделять две возможные трактовки процесса документирования: документирование как подготовка отдельного документа к публикации в бумажном или электронном виде, включающая в себя набор, корректуру, верстку, или документирование как процесс создания полного комплекса документации к ПО: технического задания, технического проекта, описания программы, руководства пользователя в бумажном и электронном виде.

В представленной системе моделей разработка технического задания (спецификации) осуществляется в ходе процесса анализа, технический проект создается в ходе процесса проектирования, описание программы может быть получено с помощью метода обратного проектирования с использованием CASE средств, необходимо только сопровождать код комментариями. Таким образом, документирование касается в первую очередь подготовки пользовательской документации.

Разработка руководства пользователя может начаться до завершения разработки приложения и завершиться одновременно с выпуском бета-версии.

Необходимо завершить подготовку файла справки до сборки бета-версии.



Процесс документирования

Модели организационных (управленческих) процессов

Управление проектом

Процесс управления проектом разработки ПО инициируется одновременно с процессом анализа в тот момент, когда заказчик подписывает договор на разработку, или отдел маркетинга принимает решение о разработке нового продукта (новых функций). Определяется название проекта, устанавливаются цели проекта, контрольные точки, границы проекта. Управление проектом включает в себя планирование проекта, определение ресурсов, отслеживание выполнения проекта (включая сроки, результаты и бюджет). Кроме того, в процессе управления проектом должны учитываться все изменения, в требованиях.

Одной из проблем управления проектами в разработке ПО является двойственность структуры работ плана. С одной стороны работы следует планировать работы в разрезе отдельных разрабатываемых процедур, классов, модулей, компонент, поскольку при оценки трудоемкости наиболее точные оценки могут быть получены в разрезе разрабатываемых компонент, с другой – необходимо планировать в разрезе функциональности, доставляемой пользователю, поскольку именно функциональность поставляется в конечном итоге заказчику. В зависимости от выбранной архитектуры и проекта, структура компонент и структура функций ПО будет отличаться между собой. Следует отметить, что в некоторых проектах (компаниях) существует тенденция свести к минимуму расхождения между составом компонент и функциональностью, такой подход лежит в основе RUP.

Планирование разработки в разрезе компонент невозможно до проектирования архитектуры, а планирование в разрезе функций возможно после проведения анализа и создания технического задания. С самого первого шага существует возможность планирования в разрезе фаз и этапов жизненного цикла.

Производительность аналитика, программиста, тестировщика зависит от следующих факторов:

- Внутренних факторов, характеризующих разработчика: квалификации, профессиональных навыков;
- Организации процесса разработки (используемых инструментов, качества получаемых материалов, методического обеспечения);
- Организации проекта - характера загрузки (например, производительность при работе над одним проектом при полной нагрузке будет выше, чем при одновременной работе над двумя проектами в режиме разделения времени);
- Организации работы: мотивации.

При планировании проектов следует учитывать, что определенная часть времени программистов, тестировщиков и других технических специалистов должна быть зарезервирована для повышения квалификации: изучения новых технологий, продуктов, причем в сфере разработки ПО время обучения может оказаться выше, чем в других видах деятельности, особенно при использовании новых технологий.

В том случае, если состав функций существенно отличается от состава компонент, а сам проект является достаточно масштабным, представляется целесообразным иметь три взаимосвязанных плана проекта: План жизненного цикла ПО, построенный на основе выбранной модели *ЖЦ ПО, План функциональности и План Компонент.*

План ЖЦ ПО содержит основные стадии и фазы проекта, контрольные точки, даты встреч с покупателем, даты поставки версий. Именно этот план используется в переговорном процессе и согласовывается с покупателем.

План функциональности содержит перечень реализуемых ПО функций: бизнес-транзакций, отчетов, возможности настройки, поиска и т.д. Этот план используется для отслеживания разрабатываемой функциональности. Между планом ЖЦ ПО и планом функциональности устанавливаются связи через выпускаемые версии. После того, как разработана архитектура, создается План разработки компонент. *План разработки компонент* создается, чтобы распределять работы между разработчиками, на основе этого плана строиться план компонентного тестирования.

План тестирования разрабатывается группой тестирования и передается менеджеру проекта для интеграции в план жизненного цикла.

Второй проблемой является согласование плана проекта и бюджета с заказчиком. Бюджет определяется менеджером проекта исходя из планируемой трудоемкости разработки требуемой функциональности, стоимости работ и нормы при-

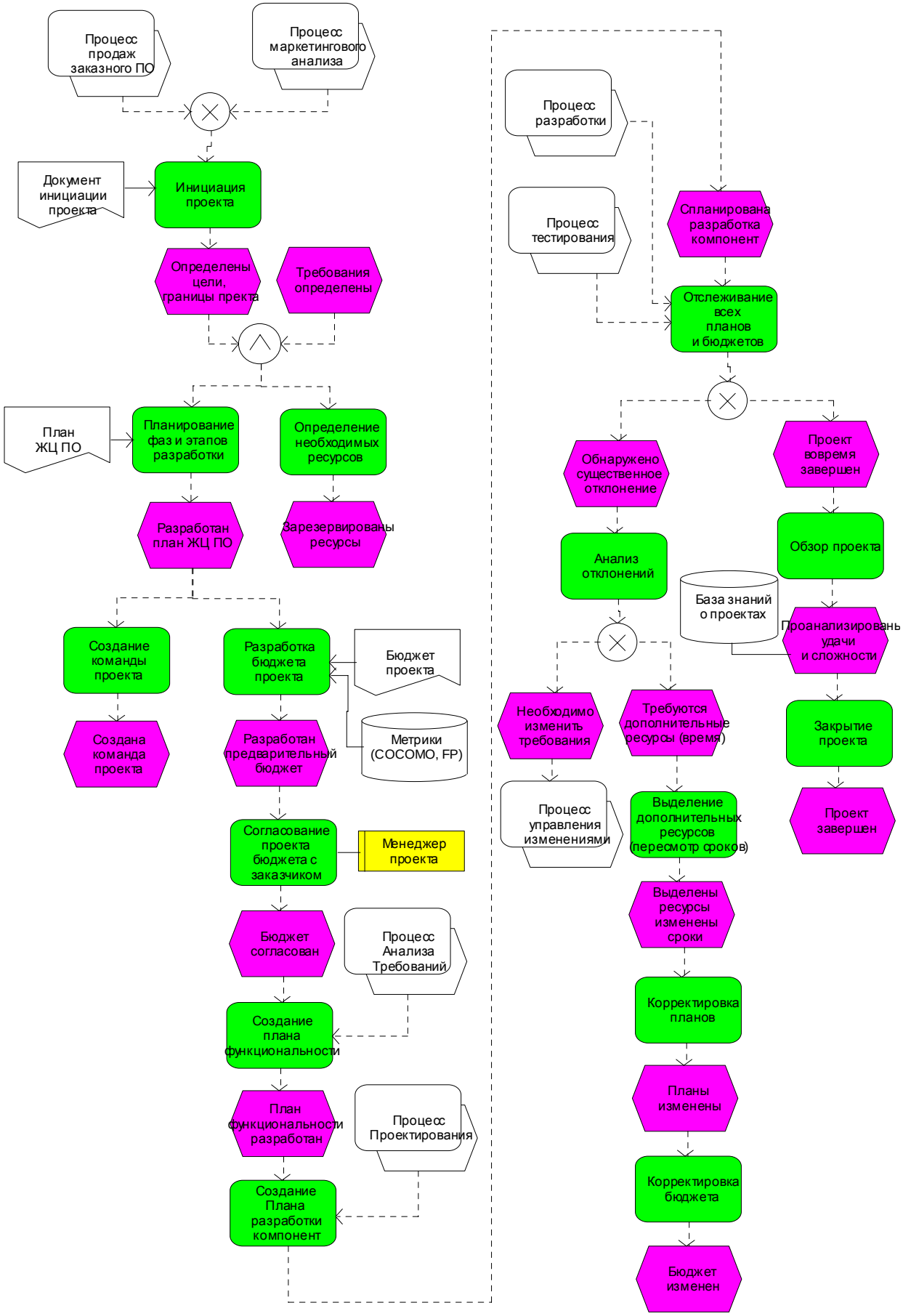
были, но, с другой стороны, бюджет представляет собой затраты заказчика и ограничен его инвестиционными возможностями по этому проекту. Проект бюджета должен быть согласован до начала процесса проектирования. В управлении проектами существуют несколько видов контрактов: контракт с твердой ценой – затраты определяются в начале проекта, контракт с возмещением затрат (с возможностью установления максимальной гарантированной выплаты). Тип контракта определяет распределение рисков между заказчиком и исполнителем.

Следует предусмотреть несколько этапов согласования бюджета, а так же процедуру корректировки бюджета в случае пересмотра планов, требований или необходимости привлечения дополнительных ресурсов.

Основными задачами управление проекта является выбор (разработка) совокупности согласованных моделей процессов разработки (планирование), а так же обеспечение следованию этим процессам (организация, контроль). Безусловно, в ежедневной работе менеджера проекта использование референтных моделей процессов в формате eEPC является неоправданно трудоемким. В действительности следует использовать шаблоны планов проекта, построенные на основе моделей процессов. Референтные модели предназначены для изменения процессов, а не для организации ежедневной работы. Безусловно, можно использовать референтные модели при создании планов (Плана ЖЦ ПО).

Предлагаемая модель процесса управления проектом отражает некоторые функции в свернутом виде, они могут быть представлены в виде соответствующей eEPC диаграммы. Например, отслеживание планов и бюджетов включает в себя отслеживание проделанной работы, контроль за израсходованными средствами, составление отчета о состоянии проекта(Project Status Report) с определенной периодичность. Функция отслеживания проекта тесно связана с управлением рисками, обычно отчет о проекте включает в себя анализ основных текущих рисков проекта и мероприятий по предотвращению этих рисков

Следует отметить, что использование данных предыдущих проектов для оценки существующей практики управления рисками возможно лишь при наличии документального подтверждения – описания результатов анализа рисков, протоколов обмана информацией о рисках, распоряжениях, направленных на уменьшение / предотвращение рисков. Поэтому можно рекомендовать всем разработчикам ПО, а особенно компаниями, занимающимся разработкой заказного ПО, разработать формы документов, содержащих информацию о рисках.



Процесс управления проектом

На наш взгляд, основным способом управления рисками является совершенствование процессов разработки ПО. Совершенствование процесса управления изменениями, создание однозначной процедуры приема/отклонения изменений позволяет уменьшить риск изменения требований. Каждое требование должно тестироваться и подвергаться количественному анализу до того, как оно будет принято к исполнению в текущей версии. Анализ требований должен осуществляться с точки зрения соответствия целям приложения, реальным возможностям их реализации (наличие необходимых ресурсов), соотношение между полезностью изменения и затратами

Управление качеством

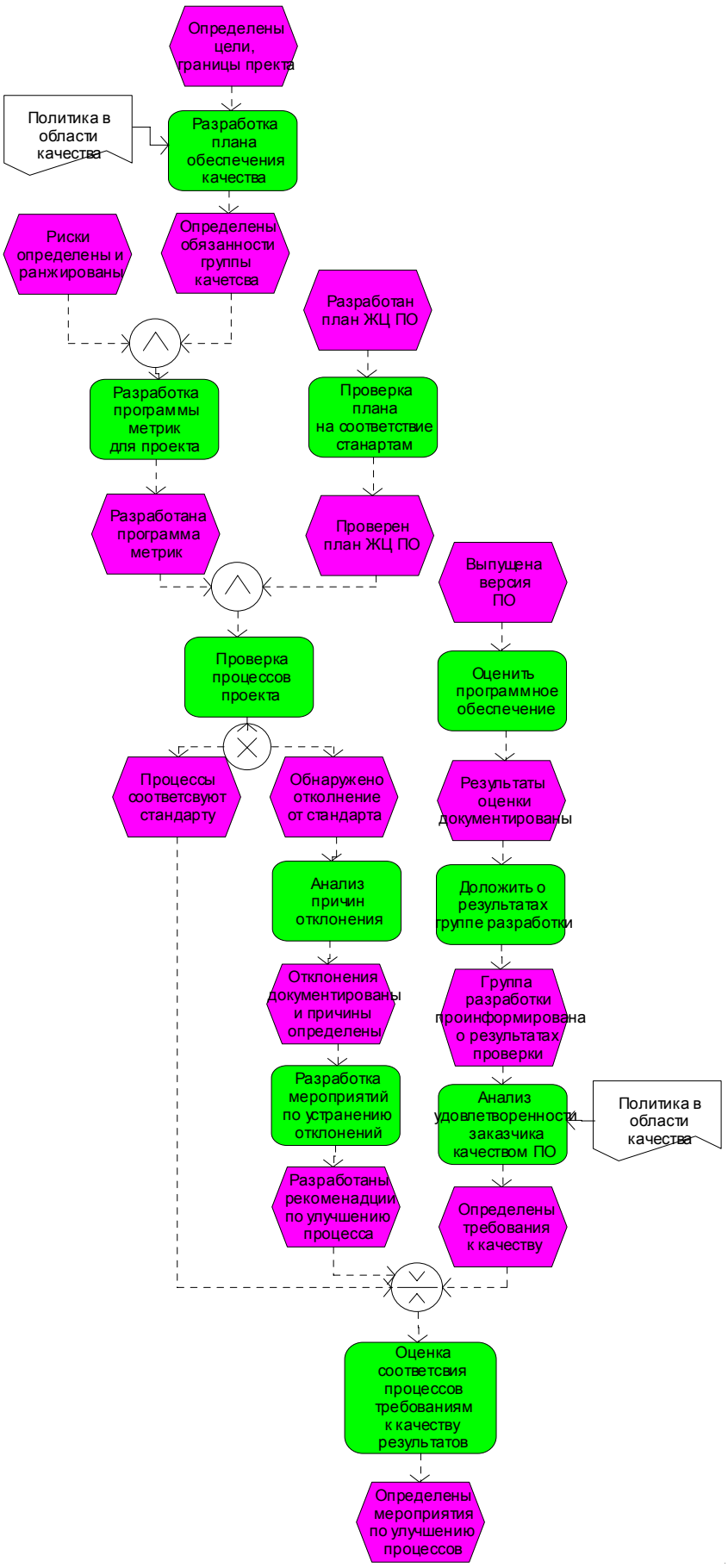
Обеспечение качества преследует цель обеспечения необходимой прозрачности процессов разработки, а управление качеством призвано обеспечить количественное понимание проектов разработки и достижение установленных целей качества. CMM раскрывает проблему управления качеством на двух уровнях: Уровень 2. Повторяемый. Обеспечение качества и Уровень 4. Оптимизирующий. Управление качеством.

В управление качеством включает в себя следующие процессы: разработка программы метрик, оценка процессов, оценка программных продуктов, выработка мероприятий по устранению отклонений, оценка степени удовлетворенности заказчика характеристиками ПО, уровнем обслуживания, оценка соответствия существующих процессам требованиям к качеству разрабатываемой продукции.

Представленная модель процессов управления качеством включает все перечисленные процессы. Модель опирается на ЖЦ ПО, процесс привязан к проекту разработки. Начало процесса управления качеством совпадает с определение целей проекта, для каждого проекта разрабатывается план обеспечения качества.

В ходе выполнения проекта группа обеспечения качества проверяет соответствие реализуемых процессов стандартам, выявляет причины отклонений. Группа обеспечения качества проводит независимую проверку разрабатываемого ПО, документации, и информирует о результатах проверок, как вышестоящий менеджмент, так и группу разработки. Кроме того, группа обеспечения качества оценивает удовлетворенность заказчика разрабатываемым ПО и документацией, предоставляемым сервисом. По результатом этой оценки уточняются требования к качеству, как для результатов конкретного проекта, так и для компании, дополняется политика качества.

Поскольку между некоторыми характеристиками качества существует обратная зависимость (например, эффективность программы и мобильность), необходимо устанавливать целевые параметры качества для проекта и их относительные приоритеты.



Процесс управления качеством

1. Capability Maturity Model® Integration (CMMI SM), Version 1.1 Continuous Representation CMU/SEI-2002-TR-011 ESC-TR-2002-011 CMMI Product Team Copyright 2002 by Carnegie Mellon University. 725 p.
2. Davenport, T.H. & Short, J.E. "The New Industrial Engineering: Information Technology and Business Process Redesign," Sloan Management Review, 1990, Summer pp. 11-27
3. Framework for Managing Process Improvement, Department of Defense: 1994. — 329 p.
4. Большая советская энциклопедия, «Советская энциклопедия» 1969 — 1978 гг.в 30 томах.
5. Брукс Ф. Мифический человеко-месяц или как создаются программные системы. – Пер. с англ. – СПб.: Символ-Плюс, 1999. – 304 с.
6. Каплан Роберт, Нортон Дейвид Стратегические карты. Трансформация нематериальных активов в материальные результаты. Олимп-бизнес, 2004 г. – 512 с
7. Ройс У. Управление проектами по созданию программного обеспечения. Унифицированный подход. М.: Лори, 2002. – 434 с.
8. Хаммер М., Чампи Дж. Реинжениринг корпорации: Манифест революции в бизнесе. Пер. с англ. /Под ред. и с предисл. В.С.Катькало. - СПб.: Издательство С. - Петербургского университета, 1997. — 332 с