Модули к системе мониторинга информационных систем

Описание процессов, обеспечивающих поддержание жизненного цикла ПО

На 20 листах

Содержание

1 Процессы, обеспечивающие поддержание жизненного цикла Модулей к системе мониторинга информационных систем

При доработке системы мониторинга использована принятая в компании методология разработки информационных систем – Rational Unified Process (RUP) – определяющая итерационный процесс разработки Модулей к системе мониторинга информационных систем.

Подробное описание принятых в Компании процессов разработки программного обеспечения и поддержания жизненного цикла ПО приведено в разделе 3 «Полное описание процессов поддержания жизненного цикла

2 Стадии

В соответствии с методологией RUP, создание Модулей к системе мониторинга информационных систем и доработка функциональности прошло в три фазы:

- Инициация,
- Исполнение,
- Завершение.

2.1 Инициация

На этом проекте, как и на других подобных работах, инициатором доработки послужили возникшие потребности производственных подразделений Заказчика в расширении функциональности системы мониторинга. После получения исходных требований исполнитель выполнил следующие работы:

- сбор всей необходимой информации;
- оценка объёма доработок;
- формирование плана разработки, тестирования и внедрения;
- оформление результатов начальной фазы – подготовка следующих документов:

1) описание новых функциональных требований;
2) план реализации со сроками;
3) оценка трудозатрат.

2.2 Исполнение проекта

Предварительные функциональные требования, сформулированные на первом этапе, были использованы в качестве исходных данных для последующего анализа.

Проектные работы включили в себя:

- детализацию и уточнение требований;
- разработку технических требований с учётом детализации и уточнения исходной информации о доработках;
- постановку технических задач исполнителям;
- тестирование новой функциональности;
- документирование.

2.3 Управление требованиями

Все поступившие от Заказчика требования, задокументированы, проанализированы и реализованы в Модулях к системе мониторинга информационных систем .

2.4 Резервирование версий

Выдаваемая Заказчику версия Модулей к системе мониторинга информационных систем, включая исходные коды и документацию, сохранена в отдельном хранилище данных. В хранилище также сохранены архивные записи мониторинга. Архивирование проводилось с целью освобождения места для хранения новых записей.

2.5 Финальная фаза проекта

Финальная фаза включила в себя сдачу ПО Заказчику и его гарантийное обслуживание.

На находящиеся в эксплуатации Модули к системе мониторинга информационных систем распространяются гарантийные обязательства Исполнителя.

3 Полное описание процессов поддержания жизненного цикла ПО

При реализации проектов в Компании используются общепринятые методологии разработки информационных систем, прежде всего Rational Unified Process (RUP) – итерационный процесс разработки.

Целью итераций является последовательное осмысление стоящих проблем, наращивание эффективности решений и снижение риска потенциальных ошибок. На выходе каждой итерации создается законченная версия работающего программного продукта.

При таком подходе можно гибко учитывать новые требования или проводить тактические изменения в деловых целях. Это позволяет на самых ранних этапах разработки выявлять и разрешать проблемы и, тем самым, снижать затраты.

Rational Unified Process предполагает разработку, реализацию и тестирование архитектуры на самых ранних стадиях выполнения проекта, устраняя самые опасные риски, связанные с архитектурой. Благодаря этому удается избежать существенных переработок на последних стадиях, если вдруг выяснится, что выбранное решение не обеспечивает, выполнение каких-либо требований.

Разработка информационных систем в компании ведется с использованием современных технологий и подходов, предназначенных как для быстрой реализации прототипов, так и для создания больших многоуровневых компонентных и масштабируемых решений корпоративного уровня. В своих проектах компания успешно применяет методологию создания прикладных программных систем, разработанную на базе RUP.

При создании приложений применяются инструментальные средства и технологии таких известных компаний как Oracle, IBM, Microsoft, Borland, а также собственные оригинальные разработки и технологии.

Для разработки больших и сложных информационных систем, создания компонентных и масштабируемых решений в компании применяются технологии JAVA и .NET. Использование собственных разработок позволяет не начинать реализацию проекта «с нуля» – имеется банк шаблонов стандартных проектов. Это дает возможность заказчику получить реально работающую и экономичную систему в максимально короткие сроки.

4 Стадии проекта

Укрупнённо проект по созданию программного обеспечения можно разделить на три фазы:

- Фаза инициации;
- Фаза исполнения;

– Финальная фаза.

4.1 Инициация проекта (начальная фаза)

Предпроектное обследование инициируется запросом от Заказчика на предоставление технико-коммерческого предложения и проводится на основе информации, которая содержится в запросе. При необходимости, Исполнитель проводит одно-два интервью с экспертами Заказчика или запрашивает дополнительные материалы. По итогам предпроектного обследования разрабатывается Коммерческое предложение, в котором представлены следующие артефакты:

– Концепция системы (функциональная и программно-аппаратная архитектура);
– План проекта;
– Оценка трудозатрат;
– Ценовое предложение.

Исполнитель использует в повседневной работе модель расчета трудоемкости и стоимости работ/услуг. Модель базируется на технологии ведения проектов, которая предполагает следующие этапы проекта: Техническое задание, Проектирование, Разработка, Тестирование, Документирование.

Такая модель оценки предназначена для расчета трудоемкости проекта на основании зависимости трудоемкости этапов проекта от трудоемкости этапа разработки и с учетом факторов влияния на различные этапы проекта.

Исходными данными для расчета трудоемкости проекта являются перечень предварительных требований к разрабатываемой системе и верхнеуровневое описание архитектуры приложения.

Расчет состоит из двух основных шагов:

Определение базовой трудоемкости этапа разработки;
Определение трудоемкости проекта.

Трудоемкость разработки является базой для определения трудоемкости остальных этапов проекта. При расчете базовой трудоемкости учитываются различные аспекты разработки, наличие которых корректирует трудоемкость данного этапа.

Для всех остальных этапов проекта определены нормативы зависимости от трудоемкости этапа Разработки (в процентах). Различные этапы могут входить или не входить в расчет трудоемкости того или иного проекта. По каждому этапу определяются факторы влияния, наличие которых увеличивает нормативную трудоемкость данного этапа.

Модель определяет три типа нормативов:

– нормативы трудоемкости разработки компонентов;

– нормативы коэффициентов для аспектов разработки;

– нормативы коэффициентов для этапов и факторов влияния.

Для расчета трудоемкости разработки определены следующие типы технических компонентов:

– пользовательский интерфейс (экранные формы, веб-страницы, отчеты, печатные формы и т. д.);

– бизнес-логика (компоненты, реализующие логику работы приложения, алгоритмы обработки данных и т. д.);

– компоненты хранения данных (таблицы БД, серверные объекты, хранимые процедуры и т. д.);

– внешний интерфейс (компоненты, реализующие логику интеграции с внешними системами).

Для каждого типа технических компонентов определены градации сложности (от **Простого** до **Очень сложного**) и определены нормы трудозатрат на разработку (в человеко-часах), основанные на экспертной оценке.

Например, для разработки экранных форм классификация компонентов по сложности будет выглядеть так:

– **Простая** — окно авторизации или окно ввода данных с проверкой вводимых данных по формату (без логических связей между различными параметрами).

– **Средняя** — форма, реализующая поиск и фильтрацию данных с постраничным выводом.

– **Сложная** — экранная форма, содержащая не более двух вкладок с проверкой ввода данных по формату и логической взаимосвязи данных.

– **Очень сложная** — экранная форма, содержащая более двух вкладок с проверкой ввода данных по формату и с учетом логических взаимосвязей.

Такая классификация является лишь иллюстрацией, окончательное отнесение компонента к той или иной категории сложности выполняет архитектор.

Норматив трудоемкости разработки определяется в человеко-часах в разрезе сложности компонентов.

Нормативы коэффициентов для аспектов разработки определяются в процентах относительно трудоемкости разработки технических компонентов и разработки в целом.

Нормативы коэффициентов трудоемкости определяются в процентном соотношении к Разработке или другим этапам и факторам влияния.

В Табл. 1 приведен пример нормативов для этапов тестирования ПО и трудоемкости управления.

Табл. 1 – Примеры коэффициентов для расчета трудоемкости этапов

Этап	Описание этапа	Детали расчета фактора/этап
Интеграционное тестирование	Интеграционное тестирование (включает время на исправление найденных во время тестирования критичных дефектов)	Процент от Разработки внешних интерфейсов
Нагрузочное тестирование	Нагрузочное тестирование (включает время на исправление найденных во время тестирования критичных дефектов)	Процент от Разработки
Документирование	Документирование	Процент от Разработки

Фактор влияния — это обстоятельство или условие, увеличивающее трудоемкость этапа или вида работы. Примеры факторов влияния и нормативы соответствующих коэффициентов приведены в Табл. 2.

Табл. 2 – Примеры коэффициентов для расчета факторов влияния

Этап	Описание этапа	Детали расчета фактора/этап
Техническое задание	Комплексно сложный продукт (необходимо написание ЧТЗ на различные компоненты/модули системы)	Процент от Технического задания
Проектирование	Использование специфической методологии, нотации, специальные требования заказчика и т. д	Процент от Проектирования
Разработка	Специфическая среда разработки (например: отсутствие средств рефакторинга и т. д)	Процент от Разработки
Разработка	Обеспечение качества: код-ревью. Этот фактор может не учитываться в случае немасштабного проекта, разработки макета или прототипа.	Процент от Разработки
Разработка	Обеспечение качества: юнит-тестирование. Этот фактор может не учитываться в случае немасштабного проекта, разработки макета или прототипа.	Процент от Разработки
Функциональное тестирование	Необходимость развертывания тестовой среды на площадке Заказчика	Процент от Функционального тестирования
Функциональное тестирование	Необходимость разработки тестовой платформы (автотесты, эмуляторы и т. д). Наличие этого фактора зависит от масштаба, критичности, планов поддержки разрабатываемой системы и т. п.	Процент от Разработки

Этап	Описание этапа	Детали расчета фактора/этап
Интеграционное тестирование	Отсутствие технических специалистов на стороне внешних систем	Процент от Интеграционного тестирования
Документирование	Написание документации на иностранном языке (перевод). За каждый дополнительный язык устанавливается свой норматив. Консультирование переводчика	Процент от Документирования
Документирование	Заказчик — Госструктура (требование оформления документации по ГОСТ или использование отраслевых или иных специализированных стандартов)	Процент от Документирования
Документирование	Необходимость подготовки учебных курсов	Процент от Документирования
Документирование	Необходимость сертификации (ФСБ, ФСТЭК, Минкомсвязь и т. д.) — полный пакет документов	Процент от Документирования
Управление	Требование ведения работ по разработке продукта на площадке Заказчика	Процент от Управления

4.2 Исполнение проекта

Общая схема технологии разработки ПО представлена на Рис. 1.

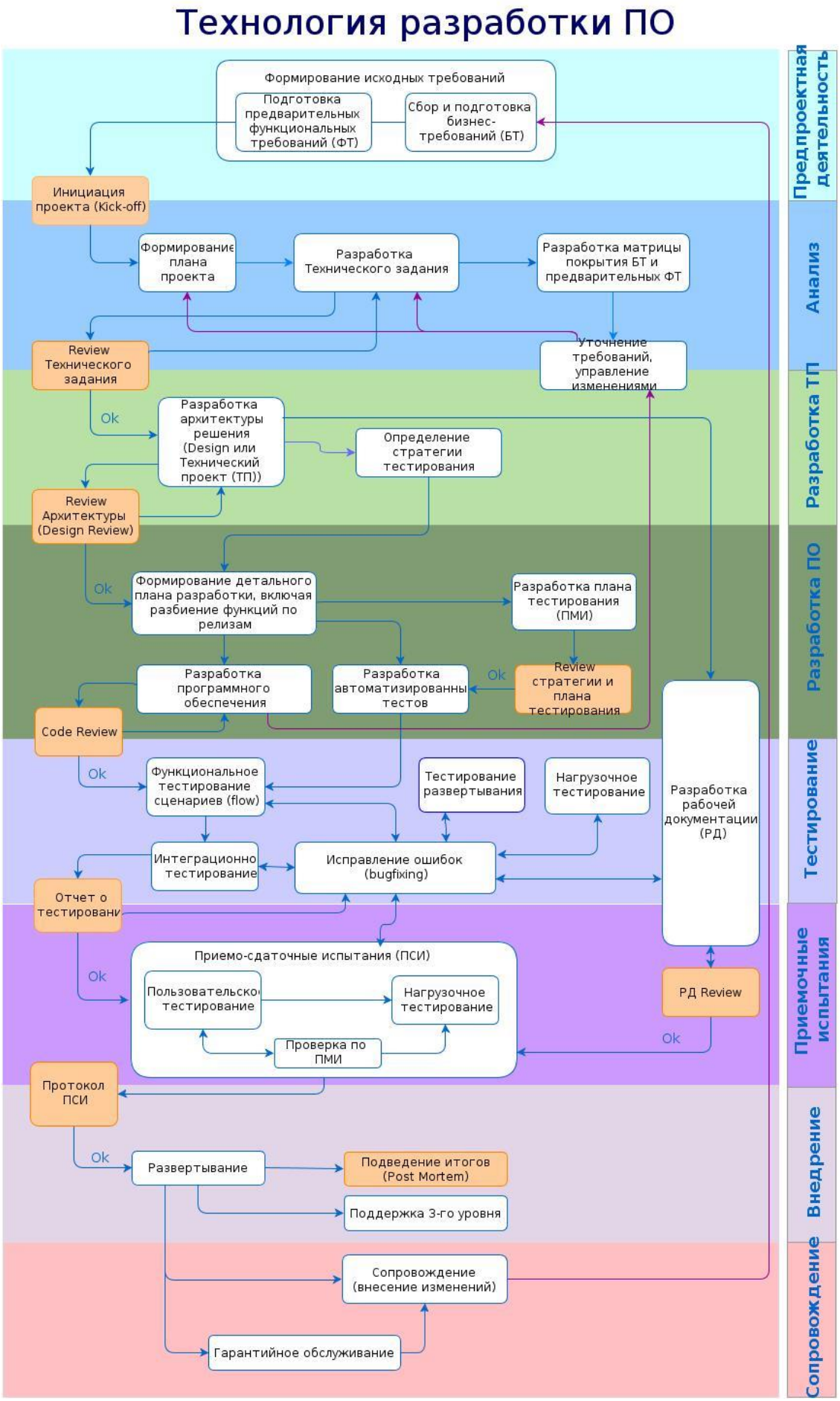

Рис. 1 – Технология разработки программного обеспечения

Разбиение на этапы приведено с учетом времени выполнения той или иной работы и не всегда совпадает с логическим разбиением работ по типам. Например, работы «Определение стратегии тестирования» и «Разработка программы и методики испытаний» относятся к внутреннему тестированию, а на схеме отнесены к этапам «Разработка технического проекта» и «Разработка программного обеспечения» соответственно.

На первом этапе, в начале проекта формируются бизнес-требования и предварительные функциональные требования, после чего инициируется проект, который переходит в этап анализа. На этапе анализа формируется детальный план проекта, проводится бизнес-анализ, разрабатываются техническое задание и матрица соответствия. Этап подразделяется на работы:

– Обследование (проведение серий интервью), детализация и конкретизация бизнес-требований;

– Разработка технического задания;

– Описание функциональности системы и бизнес-процессов «как будет». Здесь же определяется логическая структура объектов и их атрибутный состав.

По итогам этапа разрабатываются документы:

– Отчет об обследовании (необязательный документ, необходимость разработки определяется в зависимости от особенностей проекта и согласовывается с Заказчиком)

– Бизнес-требования (как правило, содержатся в запросе на технико-коммерческое предложение, в некоторых случаях требуется их детализация до разработки Технического задания);

– Техническое задание (обязательный документ).

Review Технического задания и других результатов этапа происходит на двух этапах: на этапе анализа и разработки технического проекта.

Этап технического проектирования подразделяется на работы:

– Проектирование (детализация) архитектуры разрабатываемого программного комплекса;

– Проектирование физических структур объектов хранения;

– Детальное определение интерфейсов взаимодействия с внешними системами;

– Проектирование пользовательского интерфейса.

Одновременно определяется стратегия тестирования. Результатом проектирования являются документы и модели, которые дорабатываются и корректируются в процессе реализации и оформляются на этапе документирования.

Review Технического проекта происходит и на этапе разработки программного обеспечения.

На этапе разработки программного обеспечения формируется детальный план разработки, включая разбиение функций по релизам, и разработка программного обеспечения. Кроме того на данном этапе разрабатываются автоматизированные тесты и план тестирования, которые входят и в этап внутреннего тестирования, как и review code. Детализация этапа разработки совпадает с детализацией, определяемой архитектором при расчете трудоемкости.

Внутреннее тестирование подразделяется на работы:

– Определение стратегии тестирования (на схеме отнесено к этапу «Разработка ТП»);

– Разработка программы и методики испытаний (отнесено к этапу «Разработка ПО»). Для проектов, не требующих оформления по ГОСТ, разрабатываются сценарии тестирования;

– Собственно тестирование;

– Исправление дефектов по результатам тестирования.

Отдельно, в зависимости от специфики проекта, определяется необходимость нагрузочного тестирования и автоматизированного и (или) регрессионного тестирования. Одновременно определяется необходимость разработки соответствующих скриптов.

Документирование подразумевает разработку пользовательской и эксплуатационной документации, сопровождающей поставку программного продукта, а также оформление проектной документации, разработанной в ходе проектирования в соответствии со стандартами, используемыми в проекте (ГОСТ, RUP и т. д.). Точный перечень документов, разработка которых требуется при выполнении проекта, определяется в контракте и Техническом задании. В обязательном порядке разрабатываются документы:

– Руководство пользователя;

– Руководство администратора.

Документацию можно разделить на релизную, эксплуатационную и сертификационную. Наиболее сложная и объемная документация — сертификационная.

На этапе приемочного тестирования выполняется различные виды тестирования.

4.2.1 Управление требованиями

При реализации проектов используется систематический подход к управлению требованиями и отслеживанию их изменений. В рамках процесса управления требованиями выполняется: выявление, документирование, верификация и утверждение требований, планирование реализации и отслеживание изменений требований.

Типовой состав участников проектной группы в рамках процесса управления требованиями выглядит следующим образом: системный аналитик; специалист по требованиям; рецензент требований; архитектор. Кроме того, в процессе задействован эксперт предметной области. В большинстве проектов роль эксперта, как правило, играет представитель заказчика,

но в ряде случаев эксперт предметной области входит в состав проектной группы со стороны разработчика.

– Процесс управления требованиями изображен на Рис. 2.

4.2.2 Процесс контроля версий

При разработке программного обеспечения компания придерживается ряда правил и норм, в части применения фиксированной политики релизов, резервирования исходных кодов ПО и использования централизованной системы контроля версий.

4.2.3 Политика релизов

Политика релизов (выпуск версий информационных систем) фиксируется до начала проектных работ и может отличаться в проектах, выполняемых для различных заказчиков. Поэтому, в начале проекта, прежде всего, определяется и согласовывается с заказчиком классификация выдаваемых ему релизов. Например, одной из возможных классификаций является разбиение релизов на Major/Minor-релизы с точки зрения функционала и Candidate/Final-релизы с точки зрения процесса разработки.

К Major-релизам относятся все релизы с изменениями в функциональности и релизы, содержащие пакеты исправлений выявленных критичных дефектов в функциональности промышленной версии продукта.

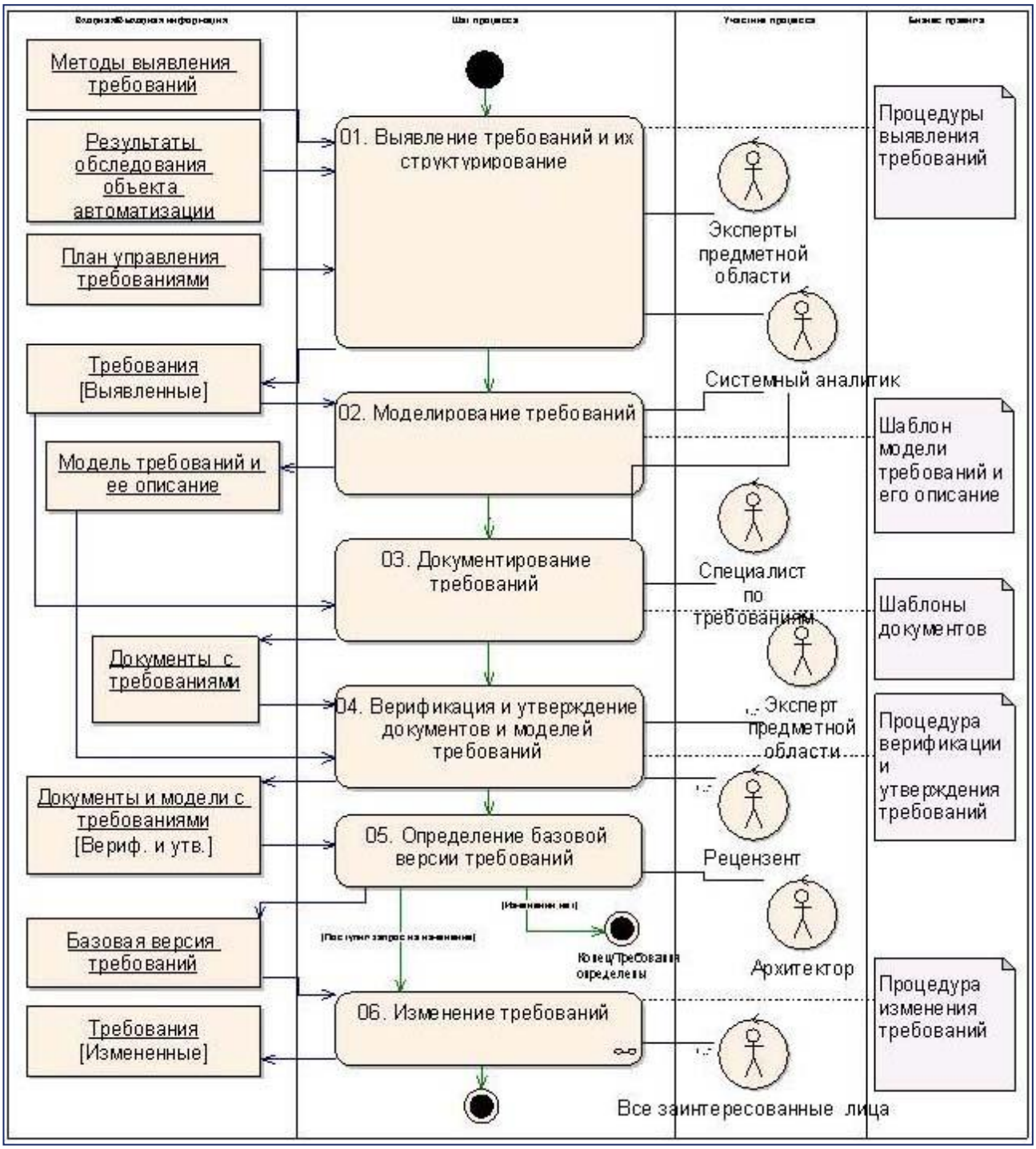

Рис. 2 – Процесс управления требованиями

К Minor-релизам относятся релизы, содержащие пакеты исправлений дефектов, обнаруженных в выданном ранее Major-релизе, не запущенном пока в промышленную эксплуатацию.

К Candidate-релизу относится готовая сборка всех компонентов продукта, содержащая реализованные функциональные требования, которые предъявляются к данному релизу, но не прошедшая этап тестирования.

После успешного прохождения этапа внутреннего тестирования релиза и выдачи его заказчику Candidate-релиз переходит в разряд Final-релизов.

4.2.3.1 Порядок именования релизов

Название любого релиза состоит из четырёх цифр X.Y.W.Z,

где:

– X – изменение платформы;

– Y – плановые релизы с новой функциональностью;

– W – внеплановые релизы с новой функциональностью, исправления продуктивной версии;

– Z – порядковый номер выданного Minor-релиза.

Если в ходе приемочного тестирования (до установки Major-релиза в продуктивную среду) были выявлены критичные дефекты, их исправление проводится в рамках Minor-релиза. Этот релиз получает метку X.Y.W, совпадающую с меткой Major-релиза, а индекс Z инкрементируется на 1.

Если ведется разработка продукта с адаптацией под различные типы платформ, в метку названия релиза вносится дополнительный параметр Platform. Соответственно, метка релиза меняется на X.Y.W.Z-Platform, где Platform — тип рабочей платформы продукта.

Если продукт состоит из отдельных модулей, подразумевающих возможность их отдельного использования или функционирования, политика их именования строго совпадает с общей принятой политикой именования продукта. Все используемые в продукте компоненты при выдаче очередной Major-версии продукта заказчику имеют одинаковые метки X.Y.W, которые гарантированно определяют принадлежность данных версий компонентов версии Major-релиза продукта, выдаваемой Заказчику.

4.2.3.2 Система контроля версий

Для работы с исходными кодами и документацией используется система управления версиями Subversion (SVN).

Такая система имеет древовидную структуру хранения данных проекта, которая позволяет:

– отслеживать все изменения;

– организовывать одновременную работу группы разработчиков над общим программным кодом продукта;

– организовывать одновременную работу над несколькими версиями продукта, восстановление измененных и удаленных частей исходных кодов продукта.

На Рис. 3 изображена схема контроля версий с использованием SVN.

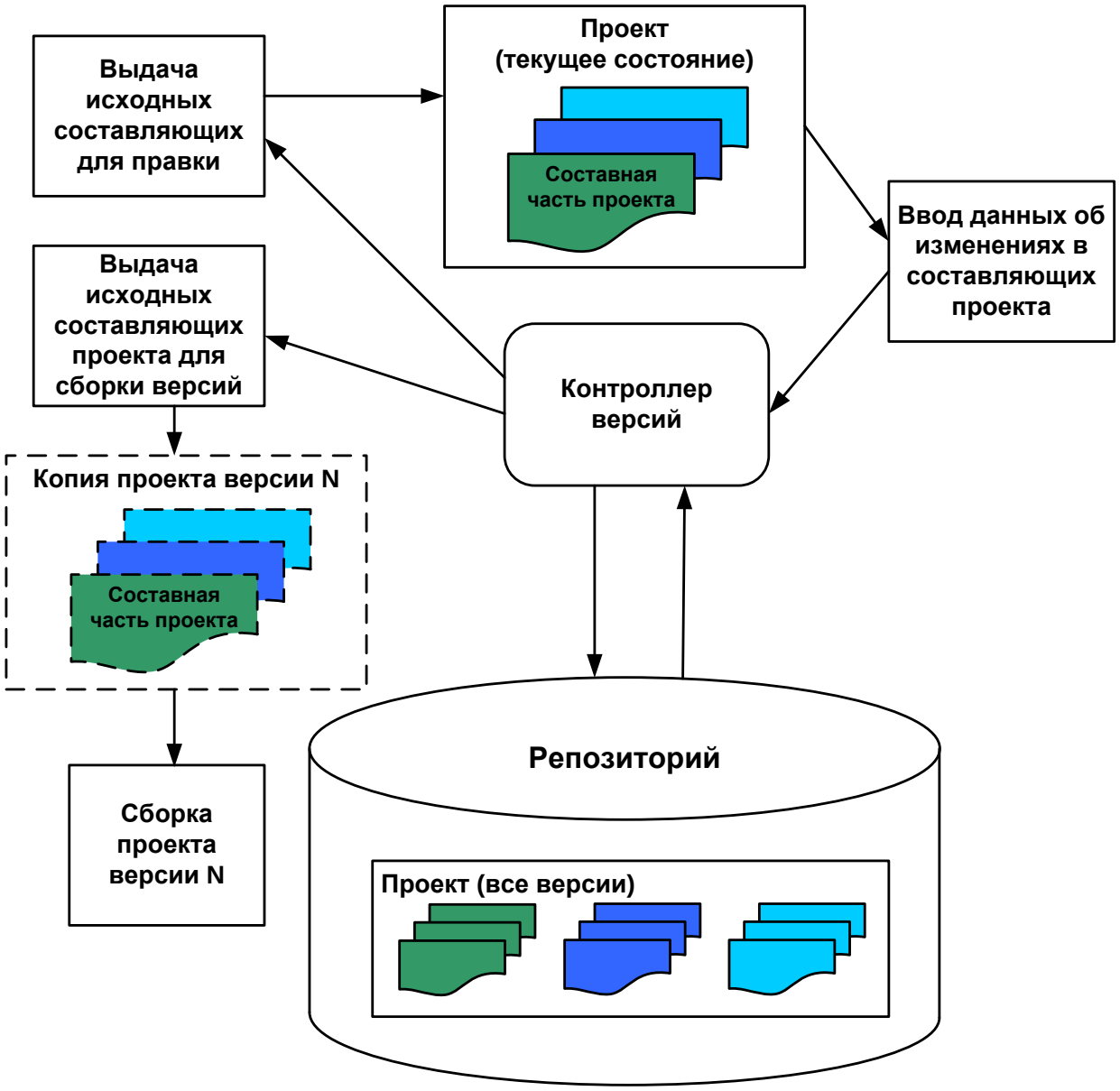

Рис. 3 – Схема контроля версий ПО

Для каждой информационной системы в SVN создается отдельный, соответствующий этой системе проект.

При работе с исходными кодами участники проекта придерживаются следующих правил SVN:

– Работа над текущей версией Major-релиза идет в стволе (HEAD) дерева проекта.

– В момент фиксации Candidate-релиза на его основе создается ветка проекта, в которой идет дальнейшая работа в рамках исправления критичных ошибок, выявленных в ходе тестирования.

– Название ветки однозначно идентифицирует разрабатываемую версию программного продукта и имеет вид: <имя проекта/компонента>X.Y.W.

– Для каждой версии Major-релиза создается ветка документации, содержащая весь перечень сопроводительных документов системы.

– Все исправления для данной ветки дерева проекта ведутся в рамках создания Minor-версий релизов продукта. При этом отдельные ветки для Minor-версий не создаются.

– Когда ветка проекта создана, ведется работа над следующей Major-версией релиза продукта.

– Все исправления, сделанные в рамках разработки Minor-версий продукта в соответствующей ветке, должны быть проанализированы на предмет их переноса в другие ветки/ствол дерева проекта — Major-версии релизов программного продукта.

Соблюдение этих правил в работе с деревом проекта, а также особенности архитектуры систем контроля версий позволяют в любой момент времени получать актуальные версии различных релизов информационных систем.

Более подробно процессы тестирования программных продуктов представлены в разделе «Тестирование ПО».

4.2.3.3 Резервирование релизов

Каждый выдаваемый заказчику релиз, включая исходные коды и документацию всех компонентов данного релиза, в обязательном порядке сохраняется в отдельном хранилище данных. Кроме того, при необходимости, создаются дополнительные резервные копии на оптических или иных носителях.

4.2.4 Тестирование ПО

Для тестирования ПО используется широкий набор программного инструментария, начиная со свободно распространяемых средств тестирования и заканчивая внутренними разработками. Так, для решения задач по автоматизации процесса тестирования, который бы позволял быстро и точно адаптироваться к специфике различных программных решений, была разработана своя тестовая платформа.

В основе этой платформы лежит технология Java, позволяющая избавиться от платформенных ограничений. Это, в свою очередь, дает возможность отделить тестовую среду от среды разработки без потери функциональности.

При реализации компонентов платформы были учтены жесткие требования к большим нагрузкам, поэтому платформа может использоваться и для функциональной, и нагрузочной серии испытаний.

Можно перечислить следующие основные преимущества разработанной в Компании тестовой платформы:

– Типизация — написание шаблонов, упрощающих написание серии тестов.

– Поддержка большинства используемых технологий посредством реализованных интерфейсов и адаптеров.

– Возможность добавления различных модулей с поддержкой дополнительных сервисов.

– Запуск тестов полностью в автоматическом режиме.

– Поддержка возможности регрессивного тестирования с расширенным протоколированием результатов.

– Поддержка нагрузочного тестирования.

– Автоматическая генерация отчетов о тестировании.

Для обеспечения качества тестирования специалистами Компании разработаны комбинации тестов новой функциональности и регрессионных тестов. Кроме функционального тестирования, периодически проводятся тесты производительности (performance), тестирование общей надежности (stability), тестирование поведения при перегрузке (stress-test), тесты пользовательского интерфейса (GUI).

4.2.5 Процесс управления дефектами

Для регистрации дефектов ПО и их исправления применяется формализованная методика.

В качестве основного инструментального средства управления дефектами используется открытое программное обеспечение Bugzilla.

Процесс управления дефектами изображен на Рис. 4.

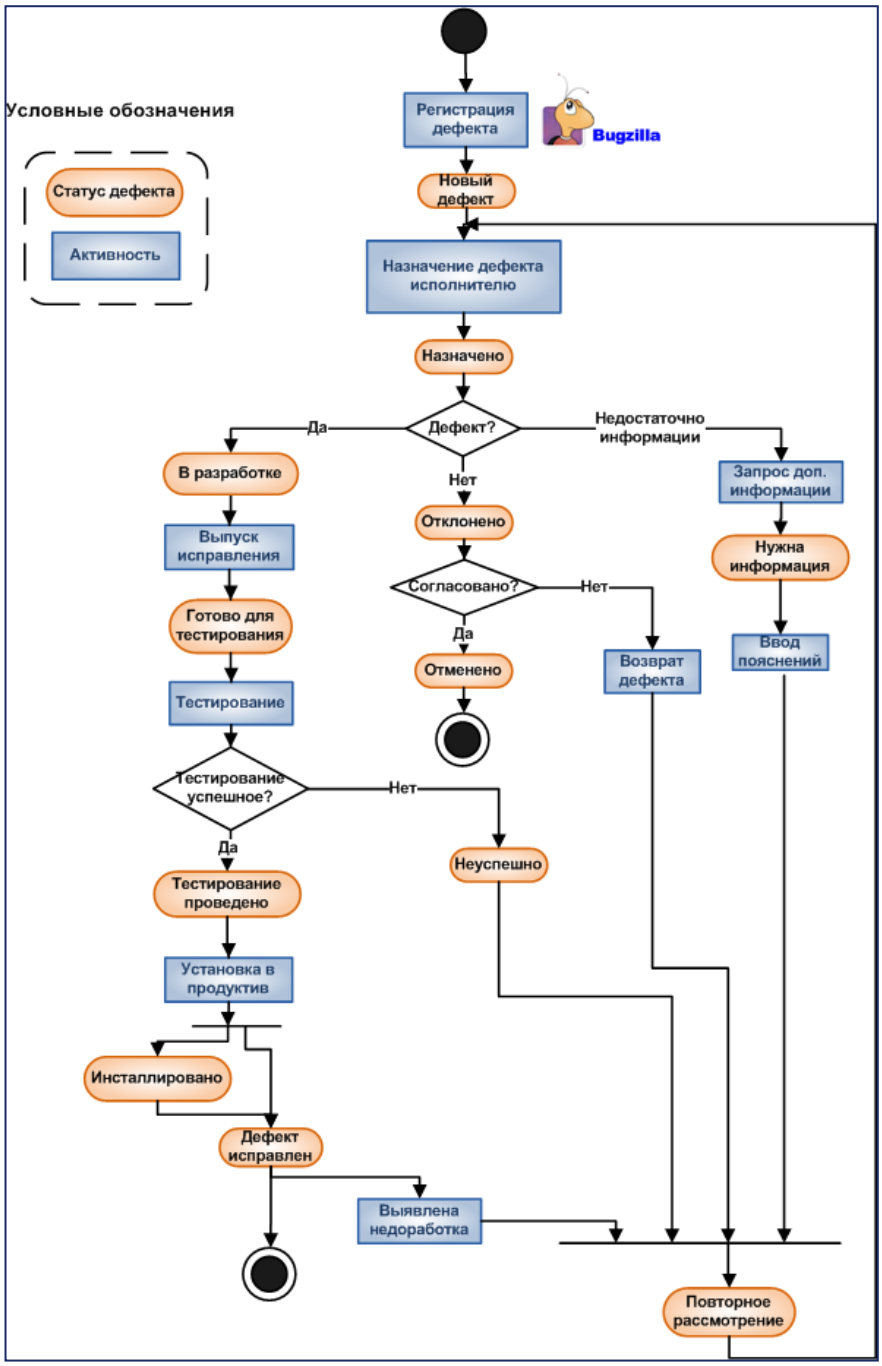

Рис. 4 – Процесс управления дефектами

4.3 Финальная фаза проекта

Финальная фаза включает в себя сдачу ПО Заказчику и его гарантийное обслуживание.

Сдачу-приёмку готового продукта принимает комиссия, состав которой утверждает Заказчик. Члены комиссии проводят испытания в соответствии с положениями и проверками, включенными в утвержденную Программу и методику испытаний. Сдача ПО оформляется протоколом приемочных испытаний и актом.

Приемо-сдаточные испытания могут проходить в два этапа. Необходимость предварительных и окончательных испытаний устанавливается по согласованию с Заказчиком.

На находящиеся в эксплуатации программные продукты распространяются гарантийные обязательства Исполнителя. Гарантийный срок сопровождения составляет 12 месяцев начиная с даты приёмки работ. По соглашению с Заказчиком этот срок может изменяться. В течение гарантийного периода все выявленные дефекты Исполнитель устраняет бесплатно.

Постгарантийное обслуживание регламентируется отдельным договором. Такой договор предусматривает заключение двухстороннего соглашения об уровне сервиса.