Традиционные процессы разработки ПО. Стадии разработки ПО. Водопадный и спиральный процессы, RUP.

Стадии разработки программного обеспечения

В компьютерной терминологии, показатель Стадии разработки программного обеспечения используется программистами для описания степени готовности программного продукта. Также стадия разработки может отражать количество реализованных функций, запланированных для определённой его версии программы.

Каждые основные версии программ в процессе своей разработки обычно проходят следующие стадии (в английском к этим названиям часто добавляют «релиз»):

Альфа — Стадия добавления новых функциональных возможностей. Программы на данной стадии могут применяться только для ознакомления с будущими возможностями.

Бета — Стадия активного бета-тестирования и отладки. Программы этого уровня могут быть использованы другими разработчиками программного обеспечения для испытания совместимости. Тем не менее программы этого этапа могут содержать достаточно большое количество ошибок.

Гамма (или Кандидат) — Стадия-кандидат на то, чтобы стать стабильной. Программы этой стадии прошли комплексное тестирование, благодаря чему были исправлены все найденные критические ошибки. Но в то же время, существует вероятность выявления ещё некоторого числа ошибок, незамеченных при тестировании.

Стабильная версия (или **Релиз**) — Стабильная версия программы, прошедшая все предыдущие стадии, в которых исправлены основные ошибки, и готовая к применению.

Эти стадии либо могут быть официально объявлены и регламентируются разработчиками, либо иногда этот термин используется неофициально для описания состояния продукта.

Итеративная разработка

Процесс итеративной (или инкрементальной) разработки стал эволюционным развитием модели водопада. Процесс состоит из серии повторяющихся итераций (их число зависит от конкретного проекта), каждая из которых фактически является полноценным мини-проектом с фазами определения требований, анализа, дизайна и т.д. В результате очередной итерации продукт приобретает новую функциональность или улучшения в существующей функциональности. Полный набор требований, зафиксированный границами проекта, оказывается реализованным после завершения финальной итерации.

Основываясь на специфике проекта и требованиях заказчика, разработчики могут выбирать, что они хотят получить в результате очередной итерации:

- Полноценную систему с ограниченной функциональностью, готовую для промышленной эксплуатации.
- Функциональные и архитектурные прототипы, непригодные для промышленной эксплуатации, но позволяющие оценить функциональный дизайн, пользовательский интерфейс, производительность и т.д.

Рисунок 2.

Итеративная разработка обладает рядом преимуществ по сравнению с последовательной моделью.

- Реализация наиболее важных функций может быть завершена в ходе нескольких первых итераций. После их завершения (то есть намного раньше окончания всего проекта) заказчик сможет начать использование системы.
- Уже в начале проекта пользователи получают возможность оценить фунциональность системы и ее соответствие своим потребностям. Необходимые изменения и дополнения могут быть сделаны в течение следующих итераций.
- Основные проектные риски могут (и должны) быть разрешены на первых итерациях. Например, архитектурное решение, приводящее к неприемлемой производительности может быть обнаружено и исправлено уже в первой итерации.

Важно понимать, что все эти преимущества проявляются только при тщательном планировании итераций, в противном случае легко получить ухудшенный вариант модели водопада.

Итеративная модель используется во многих процессах разработки, включая RUP и гибкие методологии, описанные далее в этой статье.

Водопадная модель

Модель водопада (waterfall model или последовательная разработка) – наверное, самый известный, исторически появившийся одним из первых процесс разработки. Он был описан в статье Ройса (W.W.Royce) в 1970 году (на самом деле, Ройс критиковал этот процесс, предлагая в качестве альтернативы итеративную разработку). Основная идея заключается в том, что процесс разработки делится на четко определенные фазы, выполняемые строго последовательно. Название «водопад» появилось из-за внешнего вида диаграммы, изображающей процесс:

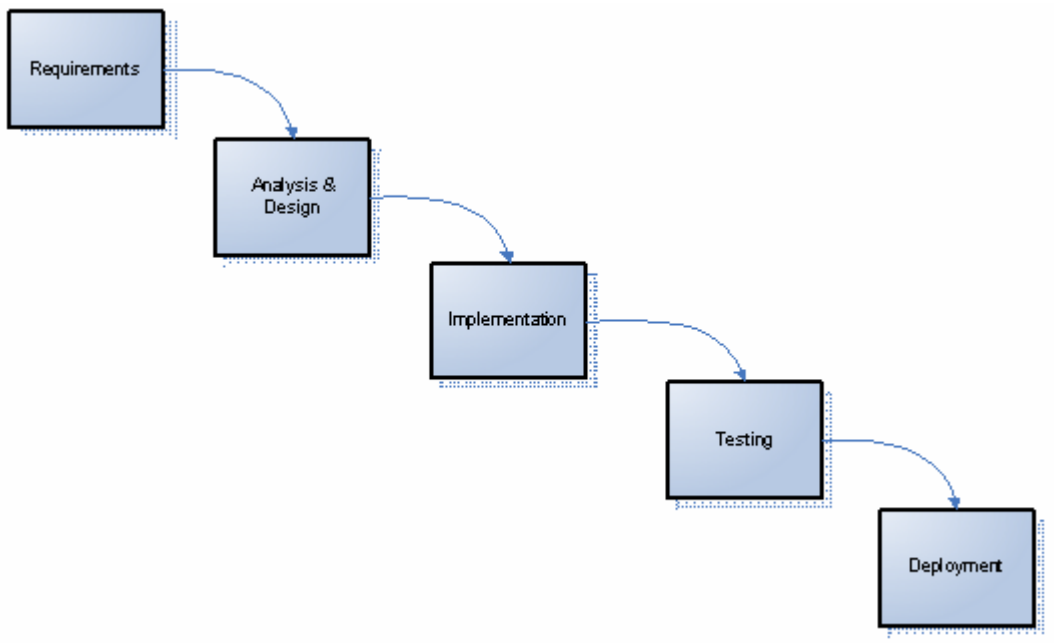


Рисунок 1.

Отдельные задачи, из которых состоит любой проект (не обязательно выполняемый по модели водопада), можно разделить между несколькими процессными областями.

Классическая водопадная модель включает следующие области:

- Разработка требований (requirements): сбор бизнес-требований заказчика и их преобразование в функциональные требования к программному продукту.
- Анализ и дизайн (analysis and design): разработка модели предметной области (domain model), проектирование схемы базы данных, объектной модели, пользовательского интерфейса и т.п.
- Реализация (implementation): создание продукта по спецификациям, разработанным на предыдущем этапе.
- Тестирование (testing): включает проверку соответствия функциональности программного продукта потребностям пользователей (validation), а также поиск дефектов в реализации.
- Развертывание (deployment): обучение пользователей, инсталляция системы, перевод в промышленную эксплуатацию.

В модели водопада каждая из процессных областей представляет собой отдельную фазу проекта. Фазы выполняются строго последовательно, т.е. анализ и дизайн начинаются после завершения разработки требований, началу реализации предшествует завершение дизайна и т.д.

Основные соображения для использования такой модели разработки вполне очевидны. Как известно, стоимость исправления ошибок, сделанных в ходе выполнения проекта зависит от того, насколько быстро эти ошибки обнаруживаются и исправляются. Ошибку в требованиях достаточно просто исправить на этапе разработки требований, но если о ней становится известно после завершения развертывания, последствия могут быть катастрофическими. Модель водопада стремится уменьшить, насколько это возможно, число таких долгоживущих ошибок. Для этого разработка дизайна не должна начинаться, пока требования не будут определены с достаточным качеством, кодирование не начинается до появления полного дизайна системы и т.д.

Насколько эффективным оказался такой подход? Он хорошо работает в проектах, где требования могут быть четко определены и зафиксированы. В таких проектах модель водопада позволяет

обеспечить заданный уровень качества (который может быть весьма высоким) и соблюдать бюджетные и временные ограничения. Благодаря этому она часто используется в больших организациях (таких как Министерство обороны США и NASA) при строгих требованиях к надежности создаваемого ПО.

Однако практика показала следующие недостатки этого процесса:

- Процесс плохо работает в проектах с нечеткими требованиями. Даже если проектная команда считает, что полностью проработала и документировала функциональный дизайн системы, он может значительно отличаться от ожиданий пользователей. С большой вероятностью это расхождение будет обнаружено не на этапе рецензирования функциональной спецификации (редкий заказчик способен представить поведение реальной системы, читая документ с описанием ее функциональности), а во время внедрения продукта.
- Процесс крайне неэффективен при постоянных изменениях требований (что как правило случается в проектах, длящихся больше одного-двух месяцев). Каждое изменение заставляет возвращаться к фазе определения требований и повторять весь процесс с начала.
- Сложно управлять рисками некоторых типов (таких, как риски, связанные с использованием новых технологий или риски некорректного определения требований). Подобные риски могут проявить себя только на этапе реализации (если не тестирования), когда число возможных путей исправления ситуации намного меньше, чем в начале проекта.
- Весьма ограничены возможности оценки и корректировки важных атрибутов проекта – скорости разработки, качества продукта, обоснованности принятых архитектурных решений. Адекватно оценить эти атрибуты становится возможным только на поздних этапах проекта.

Модель водопада является разумным выбором для типовых, стандартных проектов или при наличии жестких требований к качеству (например, при создании mission critical-систем). Тем не менее, ее недостатки весьма существенны, и для разработки коммерческого ПО, как правило, существуют значительно более эффективные альтернативы.

Спиральная модель

Спиральная модель представляет собой процесс разработки программного обеспечения сочетающий в себе как проектирование так и постадийное прототипирование с целью сочетания преимуществ восходящей и нисходящей концепции. Известная также как спиральная модель жизненного цикла является методом разработки систем, используемым в информационных технологиях. Модель сочетает в себе возможности модели прототипирования и водопадной модели. Спиральная модель ориентирована на большие, дорогостоящие и сложные проекты.

Rational Unified Process

Rational Unified Process (RUP) — методология разработки программного обеспечения, созданная компанией Rational Software.

Содержание

Принципы

В основе RUP лежат следующие основные принципы:
Ранняя идентификация и непрерывное (до окончания проекта) устранение основных рисков.
Концентрация на выполнении требований заказчиков к исполняемой программе (анализ и построение модели прецедентов).
Ожидание изменений в требованиях, проектных решениях и реализации в процессе разработки.
Компонентная архитектура, реализуемая и тестируемая на ранних стадиях проекта.
Постоянное обеспечение качества на всех этапах разработки проекта (продукта).
Работа над проектом в сплочённой команде, ключевая роль в которой принадлежит архитекторам.

Жизненный цикл разработки

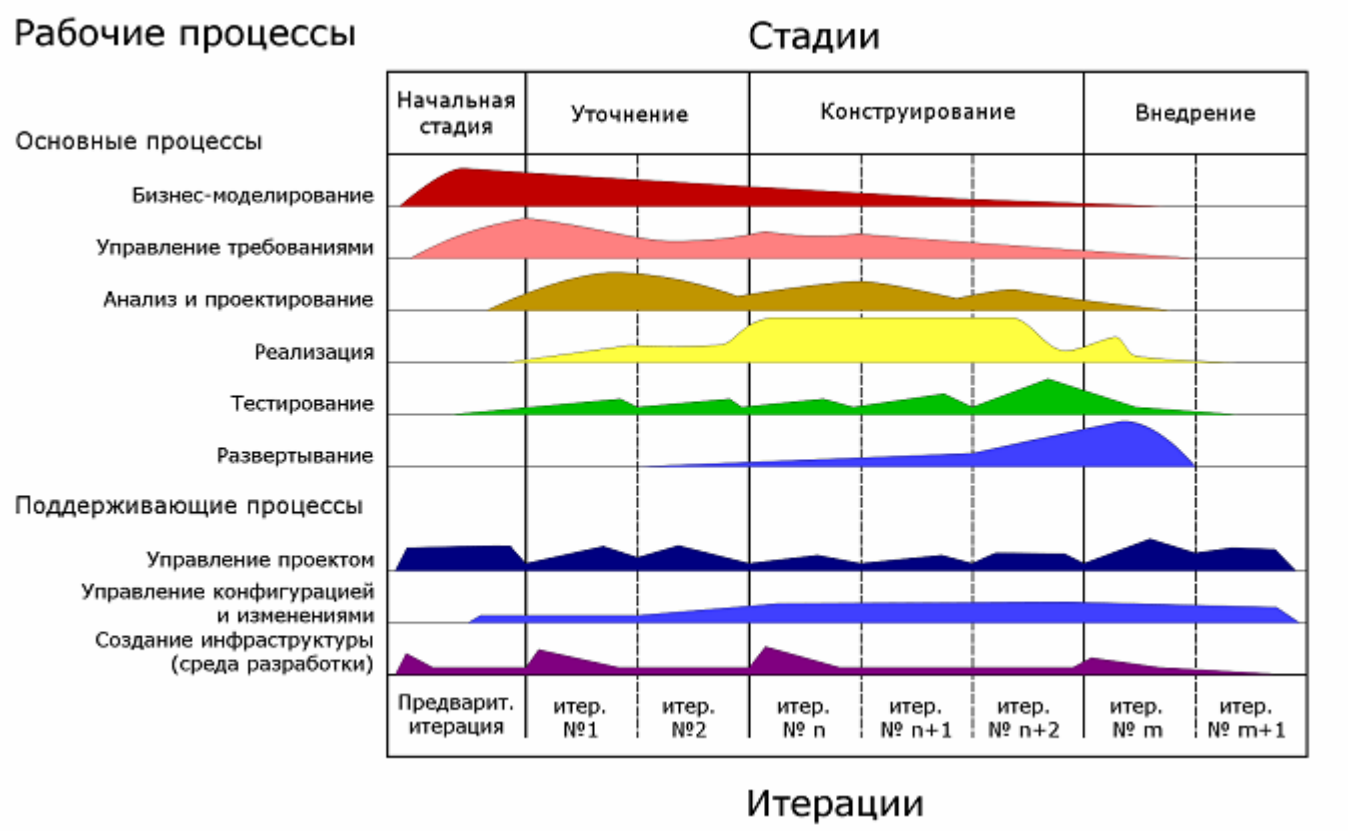
RUP использует итеративную модель разработки. В конце каждой итерации (в идеале продолжающейся от 2 до 6 недель) проектная команда должна достичь запланированных на данную итерацию целей, создать или доработать проектные артефакты и получить промежуточную, но функциональную версию конечного продукта. Итеративная разработка позволяет быстро реагировать на меняющиеся требования, обнаруживать и устранять риски на ранних стадиях проекта, а также эффективно контролировать качество создаваемого продукта.

Полный жизненный цикл разработки продукта состоит из четырех фаз, каждая из которых включает в себя одну или несколько итераций:

1. Начало (Inception)

На этом этапе:
Формируются видение и границы проекта.
Создается экономическое обоснование (business case).
Определяются основные требования, ограничения и ключевая функциональность продукта.
Создается базовая версия модели прецедентов.
Оцениваются риски.

Графическое представление процесса разработки по RUP



При завершении фазы Начало оценивается достижение вехи целей жизненного цикла Lifecycle Objective Milestone, которое предполагает соглашение заинтересованных сторон о продолжении проекта.

2. Проектирование (Elaboration)

На этапе Проектирование производится анализ предметной области и построение исполняемой архитектуры. Это включает в себя:
Документирование требований (включая детальное описание для большинства прецедентов использования).
Спроектированную, реализованную и оттестированную исполняемую архитектуру.
Обновленное экономическое обоснование и более точные оценки сроков и стоимости.
Сниженные основные риски.

Успешное выполнение фазы Проектирование означает достижение вехи архитектуры жизненного цикла (Lifecycle Architecture Milestone).

3. Построение (Construction)

Во время этой фазы происходит реализация большей части функциональности продукта. Фаза Построение завершается первым внешним релизом системы и вехой начальной функциональной готовности (Initial Operational Capability).

4. Внедрение (Transition)

Во время фазы Внедрение создается финальная версия продукта и передается от разработчика к заказчику. Это включает в себя программу бета-тестирования, обучение пользователей, а также определение качества продукта. В случае, если качество не соответствует ожиданиям пользователей или критериям, установленным в фазе Начало, фаза Внедрение повторяется снова. Выполнение всех целей означает достижение вехи готового продукта (Product Release) и завершение полного цикла разработки.