Модели процесса разработки программного продукта

В настоящее время широкое применение получают промышленные технологии создания ПП. Это разработки фирм, накопивших большой опыт создания ПО. Такие технологии представлены описаниями принципов, методов, применяемых процессов и операций и, как правило, поддерживаются набором CASE-средств (Computer-Aided Software Engineering), охватывают все этапы ЖЦ ПП и успешно применяются для решения практических задач. Рассмотрим *особенности моделей* жизненного цикла трех наиболее известных промышленных технологий.

* 1. Microsoft Solution Framework (MSF) — методология разработ-кипрограммного обеспечения фирмы «Microsoft», предназначенная для решения широкого круга задач. Технология масштабируема, т.е. настраиваема на решение задач любой сложности коллективом любой численности.
* 2. Rational Unified Process (RUP) — разработка фирмы «Rational», долгое время успешно занимавшейся созданием CASE-средств, применяемых на различных этапах жизненного цикла продукта от анализа до тестирования и документирования. Аналогично MSF технология RUP универсальна, масштабируема и настраиваема на применение в конкретных условиях.
* 3. Extreme Programming (ХР) — методология экстремального программирования, активно развивающаяся в последнее время и предназначенная для решения относительно небольших задач относительно небольшими коллективами профессиональных разработчиков в условиях жестко ограниченного времени.

**Модель Microsoft Solution Framework.**Одна из особенностей технологии MSF состоит в том, что она ориентирована не просто на создание ПП, удовлетворяющего перечисленным требованиям, а на поиск решения проблем, стоящих перед заказчиком. Как правило, предъявляемые заказчиком требования направлены на устранение некоторых глубоких проблем; и неточность, неполнота, а также изменение требований в процессе разработки — следствие их недопонимания. Поэтому в технологии MSF большое внимание уделяется анализу проблем заказчика и разработке вариантов системы для поиска их решения.

Модель жизненного цикла MSF является некоторым гибридом каскадной и спиральной моделей, сочетая простоту управления каскадной модели с гибкостью спиральной. Схема модели MSF (модели процессов) представлена на рис. 2.7. Модель жизненного цикла MSF ориентирована на *«вехи»* (milestones), т.е. ключевые точки проекта, характеризующие достижение какого-либо существенного результата. Этот результат может быть оценен и проанализирован, что подразумевает ответ на вопрос: «А достигли ли мы целей, поставленных на этом шаге?». В модели предусматривается наличие основных вех (завершение главных фаз модели) и промежуточных, отражающих внутренние этапы главных фаз.

|  |  |  |  |
| --- | --- | --- | --- |
| Развертывание  (Deploying) |  | | Создание общей |
| Решение развернуто |  | картины (Envisioning) |

|  |  |  |  |
| --- | --- | --- | --- |
|  |  | |  |
| Подтверждение готовности проекта к выпуску |  | | Утверждение документа |
|  |  | общей картины |

Стабилизация

(Stabilizing)

Планирование

(Planning)

Утверждение проектных планов

Окончательное утверждение области действия проекта

Разработка ( Developing)

Основные *фазы модели MSF.*

* 1. *Создание общей картины приложения* (Envisioning). На этом этапе решаются следующие задачи: оценка существующей ситуации; определение состава команды, структуры проекта, бизнес-целей, требований и профилей пользователей; разработка концепции решения и оценка риска. Устанавливаются две промежуточные вехи: «Организован костяк команды» и «Создана общая картина решения».
* 2. *Планирование* (Panning). Включает планирование и проектирование продукта. На основе анализа требований разрабатывается проект и основные архитектурные решения, функциональные спецификации системы, планы и календарные графики; выбираются среды разработки, тестирования и пилотной эксплуатации. Этап состоит из трех стадий: концептуальное, логическое и физическое проектирование. На стадии *концептуального* проектирования задача рассматривается с точки зрения пользовательских и бизнес-требований и заканчивается определением набора сценариев использования системы. При *логическом* проектировании задача рассматривается с точки зрения проектной команды, решение представляется в виде набора сервисов. И уже на стадии *физического* проектирования задача рассматривается с точки зрения программистов, уточняются используемые технологии и интерфейсы.
* 3. *Разработка* (Developing). Создается вариант решения проблемы в виде кода и документации очередного прототипа, включая спецификации и сценарии тестирования. Основная веха этапа — «Окончательное утверждение области действия проекта». Продукт готов к внешнему тестированию и стабилизации. Кроме того, заказчики, пользователи, сотрудники службы поддержки и сопровождения, а также ключевые участники проекта могут предварительно оценить продукт и указать все недостатки, которые нужно устранить до его поставки.
* 4. *Стабилизация* (Stabilizing). Подготовка к выпуску окончательной версии продукта, доводка его до заданного уровня качества. Здесь выполняется комплекс работ по тестированию (обнаружение и устранение дефектов), проверяется сценарий развертывания системы.
* 5. *Развертывание* (Deploying). Выполняется установка продукта и необходимых компонентов окружения, проводится его стабилизация в промышленных условиях и передача проекта в группу сопровождения, которая анализирует проект в целом на предмет уровня удовлетворенности заказчика.

**Модель Rational Unified Process**является довольно сложной, детально проработанной итеративно-инкрементной моделью с элементами каскадной модели. В модели RUP выделяются четыре основные фазы, а также девять видов деятельности (процессов). Кроме того, в модели описывается ряд практик, которые следует применять или руководствоваться для успешного выполнения проекта. RUP ориентирована на поэтапное моделирование создаваемого продукта с помощью UML (Unified Modeling Language — унифицированный язык моделирования).

Основными *фазами модели RUP* являются следующие (рис. 2.8).

* 1. *Начало проекта* (Inception). Определяются основные цели проекта, его бюджет, основные средства его выполнения: технологии, инструменты, ключевой персонал; составляются предварительные планы проекта. Основная цель этой фазы — достичь компромисса между всеми заинтересованными лицами относительно задач проекта.
* 2. *Проработка* (Elaboration). Основная цель этой фазы — на базе основных, наиболее существенных требований разработать стабильную базовую архитектуру продукта, которая позволит решать поставленные перед системой задачи и в дальнейшем будет использована как основа для разработки системы.
* 3. *Построение* (Construction). Основная цель этой фазы — детальное прояснение требований и разработка системы, удовлетворяющей им, на основе спроектированной ранее архитектуры.
* 4. *Передача* (Transition). Цель фазы — сделать систему полностью доступной конечным пользователям. Здесь происходит окончательное развертывание системы в ее рабочей среде, подгонка мелких деталей под нужды пользователей.

В рамках каждой фазы возможно проведение нескольких *итераций*, количество которых определяется сложностью выполняемого проекта.

*Основные процессы* (*деятельности*) *RUP* делятся на пять рабочих и четыре поддерживающих.

К *рабочим процессам* относятся следующие.

* 1. *Моделирование предметной области* (Business Modeling — бизнес-моделирование). Цель этой деятельности — понять бизнес-контекст, в котором должна будет работать система (и убедиться, что все заинтересованные лица понимают его одинаково), предвидеть возможные проблемы, оценить их возможные решения и последствия для бизнеса организации, в которой будет работать система.
* 2. *Определение требований* (Requirements). Цель — понять, что должна делать система, определить границы системы, основу для планирования проекта и оценок ресурсозатрат в нем.
* 3. *Анализ и проектирование* (Analysis and Design). Выработка архитектуры системы на основе ключевых требований, создание проектной модели, представленной в виде UML-диаграмм, описывающих программный продукт с различных точек зрения.
* 4. *Реализация* (Implementation). Разработка исходного кода компонентов системы, тестирование и интегрирование компонент.
* 5. *Тестирование* (Test). Общая оценка дефектов продукта и его качества в целом, оценка степени соответствия разработанного продукта исходным требованиям.

*Поддерживающими процессами* являются следующие четыре процесса.

* 1. *Развертывание* (Deployment). Цель — развернуть систему в ее рабочем окружении и оценить ее работоспособность.
* 2. *Управление конфигурациями и изменениями* (Configuration and Change Management). Определение элементов, подлежащих хранению, и правил построения из них согласованных конфигураций, поддержание целостности текущего состояния системы, проверка согласованности вносимых изменений.
* 3. *Управление проектом* (Project Management). Включает планирование, управление персоналом, обеспечение связей с другими заинтересованными лицами, управление рисками, отслеживание текущего состояния проекта.
* 4. *Управление средой проекта* (Environment). Настройка процесса под конкретный проект, выбор и смена технологий и инструментов, используемых в данном проекте.

**Модель Extreme Programming**является итерационно-инкрементной моделью быстрого создания и модификации протопопов программного продукта, которые должны удовлетворять очередному требованию (user story). Модель ХР представлена на рис. 2.9.

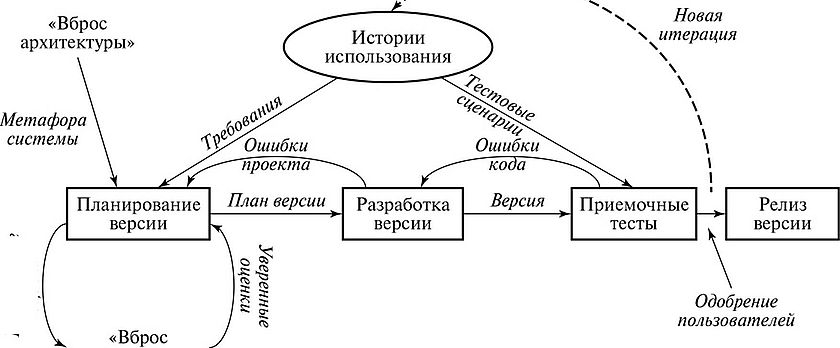
Модель ХР включает в себя выполнение следующих *основных фаз.*

* 1. *«Вброс» архитектуры —* начальный этап проекта, на котором создается видение продукта, принимаются основные решения по архитектуре и применяемым технологиям. Результатом начального этапа является метафора (metaphor) системы, которая в достаточно простом и понятном команде виде должна описывать основной механизм работы системы.
* 2. *Истории использования* (User Story) — этап сбора требований, записываемых на специальных карточках в виде сценариев выполнения отдельных функций. Истории использования являются требованиями для планирования очередной версии и разработки приемочных тестов (Acceptance tests) для ее проверки.
* 3. *Планирование версии* (релиза). Проводится на собрании с участием заказчика путем выбора User Stories, которые войдут в следующую версию. Одновременно принимаются решения, связанные с реализацией версии. Цель планирования — получение оценок того, что и как можно сделать за 1—3 недели создания следующей версии продукта.
* 4. *Разработка версии* (релиза) проводится в соответствии с планом и включает только те функции, которые были отобраны на этапе планирования.
* 5. *Тестирование версии* (релиза) проводится с участием заказчика, который ранее участвовал в составлении тестов.
* 6. *Выпуск релиза —* разработанная версия передается заказчику для использования или бета-тестирования.

По завершении цикла делается переход на следующую итерацию разработки.

*Особенности* модели жизненного цикла ХР проясняют основные принципы этого метода, и прежде всего это *принципы «живой» разработки ПО,* отраженные в *манифесте «живой» разработки ПО* люди и их общение более важны, чем процессы и инструменты; работающая программа более важна, чем исчерпывающая документация; сотрудничество с заказчиком более важно, чем обсуждение деталей контракта; отработка изменений более важна, чем следование планам.

*Основные правила модели* ХР также характеризуют ее особенности и основные техники применения:



решения»

**Рис. 2.9. Модель жизненного цикла ХР**

*Ненадежные оценки*

* • живое планирование (planning game), направленное на то, чтобы как можно быстрее определить объем работ, который нужно сделать до разработки следующей версии ПО; решение принимается на основе, в первую очередь, бизнес-приоритетов заказчика и, во-вторую, технических оценок; при этом планы изменяются, как только они начинают расходиться с действительностью или пожеланиями заказчика;
* • частая смена версий (small releases): первая работающая версия должна появиться как можно быстрее и тут же должна использоваться, следующие версии подготавливаются через достаточно короткие промежутки времени;
* • простые проектные решения (simple design): в каждый момент времени система конструируется так просто, насколько это возможно, новые функции добавляются только после ясной и обоснованной просьбы, вся лишняя сложность удаляется, как только обнаруживается;
* • разработка на основе тестирования (test-driven development) означает, что сначала пишутся тесты, демонстрирующие основные возможности системы, чтобы можно было увидеть, что система действительно заработала, а потом уже реализуются модули системы и таким образом, чтобы тесты срабатывали; при этом тесты пишутся заказчиками (заранее);
* • постоянная переработка (refactoring) системы для устранения излишней сложности, увеличения понятности кода, повышения его гибкости, при этом предпочтение отдается более элегантным и гибким решениям по сравнению с просто дающими нужный результат;
* • программирование парами (pair programming): весь код пишется двумя программистами на одном компьютере, что повышает его качество (отсутствие ошибок, понятность, читаемость);
* • постоянная интеграция (continuous integration): система собирается и проходит интеграционное тестирование как можно чаще, по несколько раз в день, каждый раз, когда заканчивается реализация очередной функции.