Нижегородский государственный университет им. Н.И. Лобачевского

Базовые принципы разработки программного обеспечения на примере Rational Unified Process

При поддержке компании Intel

Мееров И.Б., кафедра математического обеспечения ЭВМ, факультет ВМК

Содержание

- Цели лекции
- Симптомы и причины проблем в программной инженерии
- Пути решения проблем: лучшие практики программной инженерии на примере Rational Unified Process
- Обзор Rational Unified Process
  - RUP как выражение лучших практик программной инженерии
  - Структура RUP. Жизненный цикл, итерации и фазы проекта
  - Элементы RUP
- Заключение

Цели

- Определить причины возникновения проблем при разработке программного обеспечения.
- Продемонстрировать возможные пути решения указанных проблем (на примере Rational Unified Process).
- Дать введение в методологию разработки программного обеспечения (на примере Rational Unified Process).

СИМПТОМЫ И ПРИЧИНЫ ПРОБЛЕМ В ПРОГРАММНОЙ ИНЖЕНЕРИИ

Программный продукт

- Что такое программный продукт? Является ли любая программа на алгоритмическом языке программным продуктом?
- Программные проекты – разработка программных продуктов.

- **Каковы критерии успеха программного проекта?**

Критерии успеха

- Что такое программный продукт? Является ли любая программа на алгоритмическом языке программным продуктом?
- Программные проекты – разработка программных продуктов.
- Программный продукт – кому-то нужен, кем-то используется.
- Каковы критерии успеха программного проекта?
  - Проект уложился в сроки.
  - Проект уложился в бюджет.
  - Сделано то, что требовалось.
- **Всегда ли так получается?**

Виды проектов

- Критерии успеха программного проекта?
  - Проект уложился в сроки.
  - Проект уложился в бюджет.
  - Сделано то, что требовалось.
- Увы, часто программные проекты не удовлетворяют указанным критериям.
- **Виды проектов**:
  - **Успешные** (все в порядке).
  - **Испытавшие значительные трудности** (не выполнена часть критериев, но проект все-таки принес полезный результат, который применяется).
  - **Провальные** (отсутствие полезного результата).

Виды проектов. Распределение

- **Виды проектов**:
  - **Успешные** (все в порядке).
  - **Испытавшие значительные трудности** (не выполнена часть критериев, но проект все-таки принес полезный результат, который применяется).
  - **Провальные** (отсутствие полезного результата).

- **Как Вы думаете, каково распределение программных проектов между указанными группами?**

Распределение проектов по группам

Год	Провалились (%)	Трудности (%)	Успешные (%)
1994	31	53	16
1996	40	33	27
1998	28	46	26
2000	23	49	28
2004	18	53	29
2006	19	46	35
2008	24	44	32
2010	21	42	37
2012	18	43	39

* По данным Chaos report (http://www.few.vu.nl/~x/chaos/chaos.pdf, http://www.versionone.com/assets/img/files/CHAOSManifesto2013.pdf)

Глобальная проблема

- Менее половины всех проектов завершаются успешно (удовлетворяют критериям и ограничениям).
- **Позитивный симптом**: доля успешных проектов неуклонно возрастает (16% в 1994 – 39% в 2012).
- **Негативный симптом**: доля испытавших трудности проектов уменьшается медленно (53% в 1994 – 43% в 2012).

- В каких проектах возникают трудности?

Еще немного статистики

- Успешность малых проектов (**< $1 млн.**):
  - Успешные: 76%
  - Трудности: 20%
  - Провальные: 4%
- Успешность больших проектов (> **$10 млн.**):
  - Успешные: 10%
  - Трудности: 52%
  - Провальные: 38%

Цифры говорят о многом: в больших проектах индустрия далека от решения проблем. В малых виден успех современных методологий.

* По данным Chaos report (http://www.versionone.com/assets/img/files/CHAOSManifesto2013.pdf)

Некоторые симптомы возникающих проблем

- Требования заказчика не удовлетворяются.
- Требования существенно меняются в ходе проекта.
- Модули не интегрируются.
- Проект трудно поддерживать.
- Позднее обнаружение проблем.
- Плохая подготовка пользователей.
- Недостаточная производительность под нагрузкой.
- Отсутствует координация в команде.
- …

Источник: учебный курс «Object-Oriented Analysis & Design using the UML», компания Rational.

Некоторые причины возникающих проблем

- Недостаточная квалификация специалистов (менеджмент, архитекторы, аналитики, тестировщики, разработчики…).
- Неудачный выбор либо отсутствие формального процесса разработки.
- Недостаточное внимание к анализу предметной области и определению рамок проекта.
- Плохо поставленные тестирование и контроль качества.
- Недостаточное внимание к управлению изменениями и конфигурациями.
- Двусмысленные требования.
- …

Промежуточные итоги

- Индустрия решает все более сложные задачи.
- При этом возникают большие проекты (*коллективная работа, распределенная работа, задействованы большие команды специалистов разных уровней, много проблем в разных предметных областях*).
- Для того, чтобы справиться со сложностью задач, разрабатываются и совершенствуются технологии программирования и методологии разработки (*вспоминаем предыдущие лекции*).
- Усилия постепенно приводят к результату (рост доли успешных проектов).
- *В данной лекции рассматриваются некоторые признанные способы решения проблем*.

ПУТИ РЕШЕНИЯ ПРОБЛЕМ: ЛУЧШИЕ ПРАКТИКИ ПРОГРАММНОЙ ИНЖЕНЕРИИ НА ПРИМЕРЕ RATIONAL UNIFIED PROCESS

Лучшие практики программной инженерии (Rational Unified Process)

- Итеративная разработка.
- Управление требованиями.
- Использование компонентной архитектуры.
- Визуальное моделирование с использованием Unified Modeling Language (UML).
- Постоянная проверка качества.
- Управление изменениями.

Итеративная разработка

- Процесс разработки разбивается на итерации.
- Результатом каждой итерации является следующая версия системы (коды, тесты, документация…).
- На каждой итерации отрабатывается/удовлетворяется некоторая часть требований к системе.
- **Итеративная разработка** позволяет
  - обнаруживать проблемы на ранней стадии,
  - постоянно иметь работоспособный вариант системы,
  - постоянно тестировать систему,
  - адекватно измерять прогресс («где мы сейчас»),
  - положительно влиять на психологический климат в команде.

Управление требованиями

- Управление требованиями более не является «кустарной интуитивной процедурой».
  - Действует постоянно в ходе проекта.
  - Выполняется специалистами, отвечающими за этот участок работы.
  - Включает выявление, документирование, анализ, отслеживание, упорядочение по важности требований.
  - Управляется в соответствии с принятой стратегией (взаимодействие с заказчиком…).
- Ключевые вопросы: “*Solve the right problem*”, “*Build the right system*” (*OOAD using the UML, Rational*).
- Обычно изучается как отдельная дисциплина.

Использование компонентной архитектуры…

- Архитектура – ключевой элемент проекта, от которого во многом зависит его настоящее и будущее.
- Компонентная архитектура представляет ПО в виде совокупности отдельных компонентов, связанных друг с другом посредством интерфейсов.
- Компонент (*OOAD with the UML, Rational*) – нетривиальная, почти независимая, заменяемая часть системы, которая выполняет четко описанные функции в контексте хорошо определенной архитектуры.
- Компонентная архитектура обеспечивает гибкость к изменениям, расширяемость, повторное использование, упрощает коллективную разработку, отладку, тестирование, документирование…

Использование компонентной архитектуры

- Компонентная архитектура обеспечивает гибкость к изменениям, расширяемость, повторное использование, упрощает коллективную разработку, отладку, тестирование, документирование…
- Компонентная архитектура допускает повторное использование как программных решений, так и архитектурных решений.

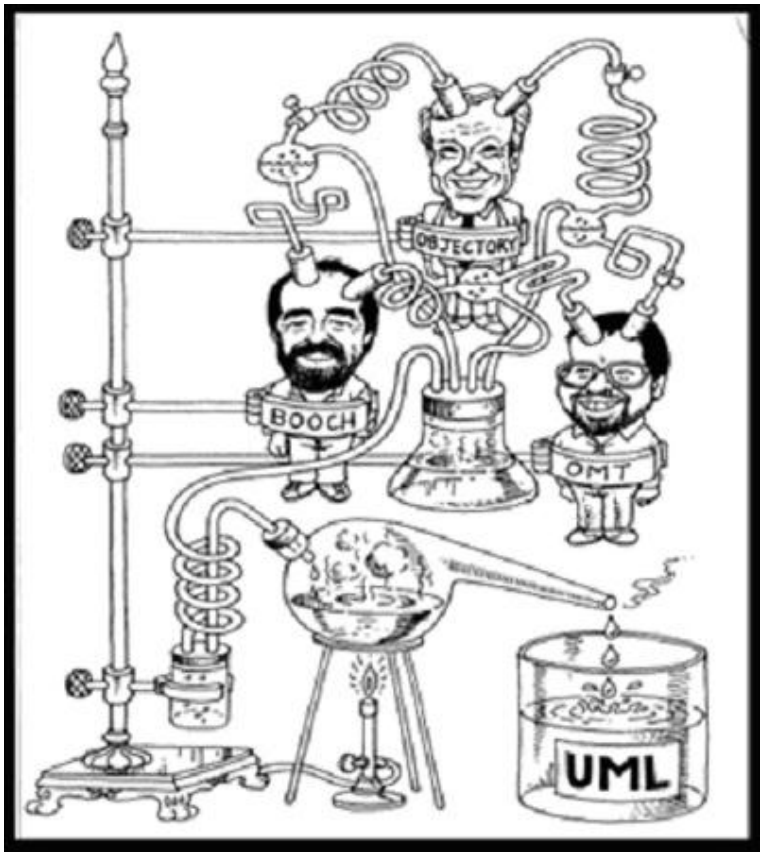
- Основные идеи базируются на принципах или являются дальнейшим развитием идей объектного подхода.

Визуальное моделирование с использованием Unified Modeling Language

- Зачем нужно визуальное моделирование?
  - Исключить двусмысленность формулировок.
  - Упростить понимание обсуждаемых идей и подходов.
  - Создать базис для согласованной работы коллектива и документирования принятых проектных решений.
  - Концентрировать внимание, рассматривая проблему на нужном уровне детализации.
  - Наглядно отобразить, в какой степени отдельные элементы решения соотносятся друг с другом.

Unified Modeling Language

- UML – Unified Modeling Language, унифицированный язык для визуального моделирования, результат объединения многих идей и принципов (Booch, Rumbaugh, Jacobson).
- Развивается около 20 лет, сейчас является общепринятым (UML 2.x).
- Применяется для проектирования и документирования программных систем.



Three Amigos, How it all began

Источник:
http://www.ibm.com/developerworks/rational/library/998.html

Постоянная проверка качества

- Управление качеством – специальная дисциплина.
- Выполняется непрерывно в ходе проекта.
- Позволяет выявлять ошибки и проблемы на ранней стадии.
- Приводит к снижению ущерба от ошибок.
- Позволяет вовремя принять меры и оптимизировать работу, выделить дополнительные ресурсы при необходимости.
- Контролирует
  - функциональность,
  - производительность,
  - надежность.

Управление изменениями

- Выполняется непрерывно в ходе проекта.
- Создает для каждого разработчика среду, максимально изолированную от изменений, не имеющих к нему прямого отношения.
- Обеспечивает автоматизацию интеграции модулей и сборки.
- Управляет совместной деятельностью всех разработчиков.
- Исключает хаос (снижает риск возникновения) при условии одновременной работы множества разработчиков, представляющих разные команды, расположенные в разных местах, использующих разное программное и аппаратное обеспечение.

Взаимосвязь лучших практик

- Лучшие практики поддерживают и усиливают друг друга.
- Должны использоваться совместно.

- “Дьявол кроется в деталях” – многие хорошие идеи не приносят пользы из-за неудачной реализации.
  - Необходимо внимательное изучение кратко рассмотренных аспектов.
  - В данном курсе – краткий обзор и, частично, практическое освоение.
  - Каждая из перечисленных практик – тема отдельного курса.

ОБЗОР
RATIONAL UNIFIED PROCESS

RUP как выражение лучших практик программной инженерии

- Rational Unified Process – процесс разработки ПО.
  - Предполагает использование объектного подхода.
  - Является четким руководством для членов команды.
  - Ориентирован на итеративную разработку.
  - Основан на вариантах использования и компонентной архитектуре.
  - Использует UML в качестве единого языка для документирования (нотация, модели, диаграммы).
  - Является интегрированным продуктом (поддержан специализированным ПО).
  - Допускает гибкую настройку и изменения под конкретные нужды.

Базовые принципы RUP*

- Непрерывно ведите наступление на риски.
- Обеспечьте выполнение требований заказчиков.
- Сконцентрируйтесь на исполняемой программе.
- Приспосабливайтесь к изменениям с начала проекта.
- Рано закладывайте исполняемую архитектуру.
- Стройте систему из компонентов.
- Работайте вместе как одна команда.
- Сделайте качество образом жизни.

** Кролл П., Крачтен Ф. Rational Unified Process – это легко. Руководство по RUP для практиков*

Структура RUP. Жизненный цикл, итерации и фазы проекта

RUP в одном слайде*:

Источник: www.wikipedia.org

Элементы RUP и связанная терминология

- Стадии:
  - Inception (Начало).
  - Elaboration (Проектирование, Уточнение).
  - Construction (Конструирование, Построение).
  - Transition (Внедрение).
- Итерации.
- Вехи (milestones).
- Рабочие процессы (workflows).
- Модели.
- Роли.

RUP по стадиям…

- **Inception**: определение рамок проекта.
  - Что будет включено в продукт, что не будет включено.
  - Бизнес-план – привлечение ресурсов в проект, экономическое обоснование.
  - Идентификация основных Актеров (actor) и Основных вариантов использования (Use case).
- **Elaboration**: планирование проекта, разработка архитектуры, определение функциональности.
- **Construction**: создание продукта.
- **Transition**: поставка заказчику.

RUP по стадиям…

- **Inception**: определение рамок проекта.
- **Elaboration**: планирование проекта, разработка архитектуры, определение функциональности.
  - Определиться с основными требованиями.
  - Разработать архитектуру проекта.
  - Свести к минимуму риски.
  - Понять стоимость разработки системы.
- **Construction**: создание продукта.
- **Transition**: поставка заказчику.

RUP по стадиям…

- **Inception**: определение рамок проекта.
- **Elaboration**: планирование проекта, разработка архитектуры, определение функциональности.
- **Construction**: создание продукта.
  - Итеративно создать beta-версию продукта.
- **Transition**: поставка заказчику.

RUP по стадиям

- **Inception**: определение рамок проекта.
- **Elaboration**: планирование проекта, разработка архитектуры, определение функциональности.
- **Construction**: создание продукта.
- **Transition**: поставка заказчику.
  - Поставить заказчику.
  - Развернуть на инфраструктуре заказчика.
  - Выполнить доработку до релиза.
  - Обучить пользователей.

Фазы, итерации, вехи

- Каждая фаза может включать одну или несколько итераций – по мере необходимости, в зависимости от проекта.
- Каждая фаза завершается главной вехой (major milestone).
- Каждая итерация порождает работоспособный релиз с частичной функциональностью.
- Каждая итерация завершается промежуточной вехой.

- Вехи – контрольные точки. Устанавливают цели, позволяют понять, достигнут ли результат, можно ли двигаться дальше.

Элементы RUP

- **Роли** – кто выполняет работу.
- **Задачи** (действия) – в чем состоит работа.
- **Артефакты** – что является результатом работы.
- **Рабочие процессы** – каким способом достигать результата. Строят разные модели.
  - Бизнес-моделирование.
  - Управление требованиями.
  - Анализ и проектирование.
  - Реализация.
  - Тестирование.
  - Развертывание.

Структура RUP. Жизненный цикл, итерации и фазы проекта

И снова RUP в одном слайде*:

Источник: www.wikipedia.org

Заключение

- Рассмотрены некоторые проблемы разработки ПО и причины их возникновения.
- Изучены лучшие практики программной инженерии (на основе RUP), показано, как они помогают избегать многих проблем при разработке ПО.
- Сделан обзор RUP – одного из процессов разработки ПО, показано, как RUP реализует лучшие практики программной инженерии.

Бороться не с последствиями, а с причинами проблем – основная идея данной лекции.

Контрольные вопросы/вопросы для обсуждения

- Назовите известные вам симптомы того, что что-то в разработке программного проекта пошло не так. Сформулируйте причины возникших трудностей.
- Перечислите лучшие практики программной инженерии с точки зрения разработчиков RUP. Поясните кратко основные идеи, заложенные в данные практики.
- Назовите основные составляющие RUP.
- Сформулируйте, как RUP реализует лучшие практики программной инженерии.
- Как вы думаете, какова роль методологии при разработке ПО? В какой степени она влияет на результат? Зависит ли степень влияния от размера проекта, если да, то как?

Литература к лекции

1. **Кролл П., Крачтен Ф.** Rational Unified Process – это легко. Руководство по RUP для практиков. – Изд-во “КУДИЦ-Образ”, 2004.
2. **Буч Г., Рамбо Д., Джекобсон И.** Язык UML. Руководство пользователя. – Изд-во “ДМК Пресс”, 2007.
3. **Соммервиль И.** Инженерия программного обеспечения. – М.: Издательский дом "Вильямс", 2002. – 624 с.
4. **Буч Г.** Объектно-ориентированный анализ и проектирование с примерами приложений на C++. Второе издание. – Бином, 1998.

При подготовке к лекции использовались материалы курса «Object-Oriented Analysis & Design using the UML», разработанного компанией Rational.