Технология объектно-ориентированного проектирования ИС (разработки программного обеспечения) – Rational Unified Process (RUP)

д.э.н., проф. Тельнов Ю.Ф.

Технологии проектирования ИС (разработки ПО)

Каскадная модель

Предложена У.У. Ройсом в 1970 г.

Предусматривает строгую последовательность этапов разработки, при которой переход к следующему этапу осуществляется только после того, как будут завершены все задачи предыдущего.

Спиральная модель

Предложена Б. Боэмом в 1988 г. Модель сочетает в себе возможности каскадной модели разработки и постадийного прототипирования. Каждый виток спирали предполагает выпуск очередной версии системы, уточнение задач проекта. Переход к следующей стадии проекта осуществляется в соответствии с планом работ, даже, если не вся работа на предыдущей стадии завершена.

IBM Rational Unified Process

Итеративный процесс разработки программного обеспечения, созданный компанией Rational Software в 1996 г.

Microsoft Solutions Framework

Методология разработки программного обеспечения, предложенная Microsoft в 1994 г.

Agile

Концептуальный каркас гибкой методологии разработки ПО. Нацелен на раннюю минимизацию проектных рисков путём сведения всего проекта разработки к серии повторяемых циклов, называемых итерациями. Каждая итерация — это мини-проект, результатом которого является новая версия продукта. Большое внимание уделяется непосредственному общению с заказчиком.

Манифест Agile-методов был принят в 2001 г.

XP

Методология «экстремального программирования». Предполагает короткий цикл обратной связи и непрерывный процесс разработки

Scrum

Методология управления Agile-разработкой. Делает акцент на качественном контроле процесса разработки

OpenUP

Итеративно-инкрементальный метод разработки ПО, рассматриваемый как легкий и гибкий вариант RUP

Rapid Application Development

Концепция создания средств разработки программных продуктов, уделяющая внимание быстроте и удобству программирования

V-Model

Вариация каскадной модели, в которой задачи разработки идут сверху вниз по левой стороне буквы V, а задачи тестирования — вверх по правой стороне буквы V

Учебный график

Неделя	Лекции	Практические занятия	Контрольные мероприятия
Неделя 1	Общая характеристика технологии RUP	Уточнение и спецификация требований (Use-Case). Задание на лабораторную работу 1	
Неделя 2	Начальная фаза разработки ИС	Уточнение и спецификация требований (Use-Case).	
Неделя 3	Фаза проектирования ИС	Уточнение и спецификация требований (Use-Case).	Защита первой части лабораторной работы 1
Неделя 4	Фаза построения ИС (реализация)	Проектирование диаграммы активностей	
Неделя 5	Фаза внедрения ИС	Проектирование диаграммы активностей	
Неделя 6	Технологии гибкого проектирования ИС	Проектирование диаграммы классов	
Неделя 7	Студенческая конференция – Технологии проектирования ИС	Проектирование диаграммы классов	
Неделя 8	Обзорная лекция Тестирование	Проектирование диаграммы классов	Защита второй части лабораторной работы 1

Вопросы

1. Сущность, особенности подхода, архитектура RUP
2. Динамическая структура RUP, фазы разработки
3. Статическая структура RUP, процессы разработки

Литература

- Н.Н. Заботина. Проектирование информационных систем. – М.: Инфра-М, 2011
- П. Кролл, Ф. Крачтен. Rational Unified Process – это легко. Руководство по RUP для практиков. – Кудиц-Образ, 2004.
- Р. Деннис Гиббс. Управление проектами с помощью IBM Rational Unified Process Кудиц-Пресс, 2007.
- Крачтен.Ф. Введение в Rational Unified Process. Изд. 2-е.- М.: Издательский дом "Вильямс", 2002. - 240 с.: ил.
- Якобсон А., Буч Г., Рамбо Дж. Унифицированный процесс разработки программного обеспечения - Спб.: Питер, 2002. - 496 с.: ил.
- Фаулер М., Скотт К. UML в кратком изложении. Применение стандартного языка объектного моделирования: Пер. с англ. – М.:Мир, 1999. – 191 с., ил.
- А.М. Вендров. Проектирование программного обеспечения экономической информационной системы. – М.: Финансы и статистика, 2000.
- Г.Н. Смирнова, А.А. Сорокин, Ю.Ф. Тельнов. Проектирование экономических информационных систем. - М.: Финансы и статистика, 2003

1. Общая характеристика Rational Unified Process - RUP

- Четко-определенная и структурированная технология программной инженерии, поддерживающая полный жизненный цикл проекта
- Документированный набор наилучших практических методов для разработки ПО
- Технологический продукт, настраиваемый на особенности автоматизации конкретного объекта.
- Используется для успешной разработки больших и сложных программных систем

Базовые принципы RUP (до 2005 г.)

1. **Разрабатывайте итеративно - Controlled Iterative.** Прототипная спиральная разработка, главное работающий правильно и эффективно программный код. Этот подход обеспечивает большую гибкость при изменяющихся требованиях и тактических коррективах в бизнес-целях, что позволяет более эффективно и заблаговременно идентифицировать и снижать проектные риски.
2. **Управляйте требованиями - Use-case driven.** Обеспечьте выполнение требований заказчиков -документирование и строгое исполнение требований, между всеми участниками проекта обеспечивать единое понимание ожидаемых функциональных возможностей,
3. **Используйте компонентную архитектуру - Architecture Centric.** Стройте систему из компонентов – использование повторно-используемых компонентов (объектно-ориентированный подход)_ Проектирование, реализация и тестирование архитектуры системы на ранних стадиях использование.
4. **Моделируйте визуально - Visual Modeling Techniques.** Моделирование по методологии RUP является объектно-ориентированным и базируется на понятиях объектов, классов и зависимостей между ними с помощью унифицированного языка моделирования **Unified Modelling Language™ (UML)**
5. **Непрерывно проверяйте качество – Continuos Quality Management .** Сделайте качество образом жизни, а не запоздалой идеей - Управление качеством (тестирование на всех фазах, а не только в конце развертывания). Оценка качества всех работ, выполняемых любыми участниками проекта, использует объективные метрики и критерии. Методология RUP создавалась с прицелом на поддержку управления качеством в рамках требований SEI CMM/CMMI.
6. **Управление изменениями** с самого начала разработки и на всех фазах ЖЦ - **Requirements Configuration and Change Management.** Приспосабливайтесь к изменениям с самого начала проекта - Закладывание основу используемой архитектуры как можно раньше
7. **Работайте вместе как одна команда** - Командный принцип разработки (архитекторы, аналитики, программисты, тестировщики в одном проекте)

Шесть лучших практик RUP (после 2005 г.)

- **Практика 1. Приспосабливайте процесс –** Сильная формализация процесса проектирования связана с распределенностью и сложностью проекта:
  - Географическое распределение проектной команды
  - Географическое распределение большого числа пользователей
  - Сложная структура проекта, множество подрядчиков
  - Жесткое соответствие стандартам
  - Техническая сложность программного продукта
  - Завершающая стадия проекта (в начале проекта формализация меньшая)

  При разработке нескольких проектов накопление проектных решений в корпоративной памяти (репозитории)
- **Практика 2. Ищите компромисс для разных приоритетов заинтересованных лиц** - избегать заказной разработки там, где возможно, использование коммерческих продуктов (Commercial Off-The-Shelf –COTS), сервисов и компонентов повторного использования: снижение затрат и рисков
- **Практика 3. Организуйте совместную работу команд** как в организации – заказчике, так и в организации-подрядчике
- **Практика 4. Показывайте результат итеративно**
  - Декомпозиция функциональности
  - Демонстрация прототипов
  - Создание прототипов на ранних стадиях
  - Результат итерации – новая версия
  - Измерение реального прогресса по итогам итераций

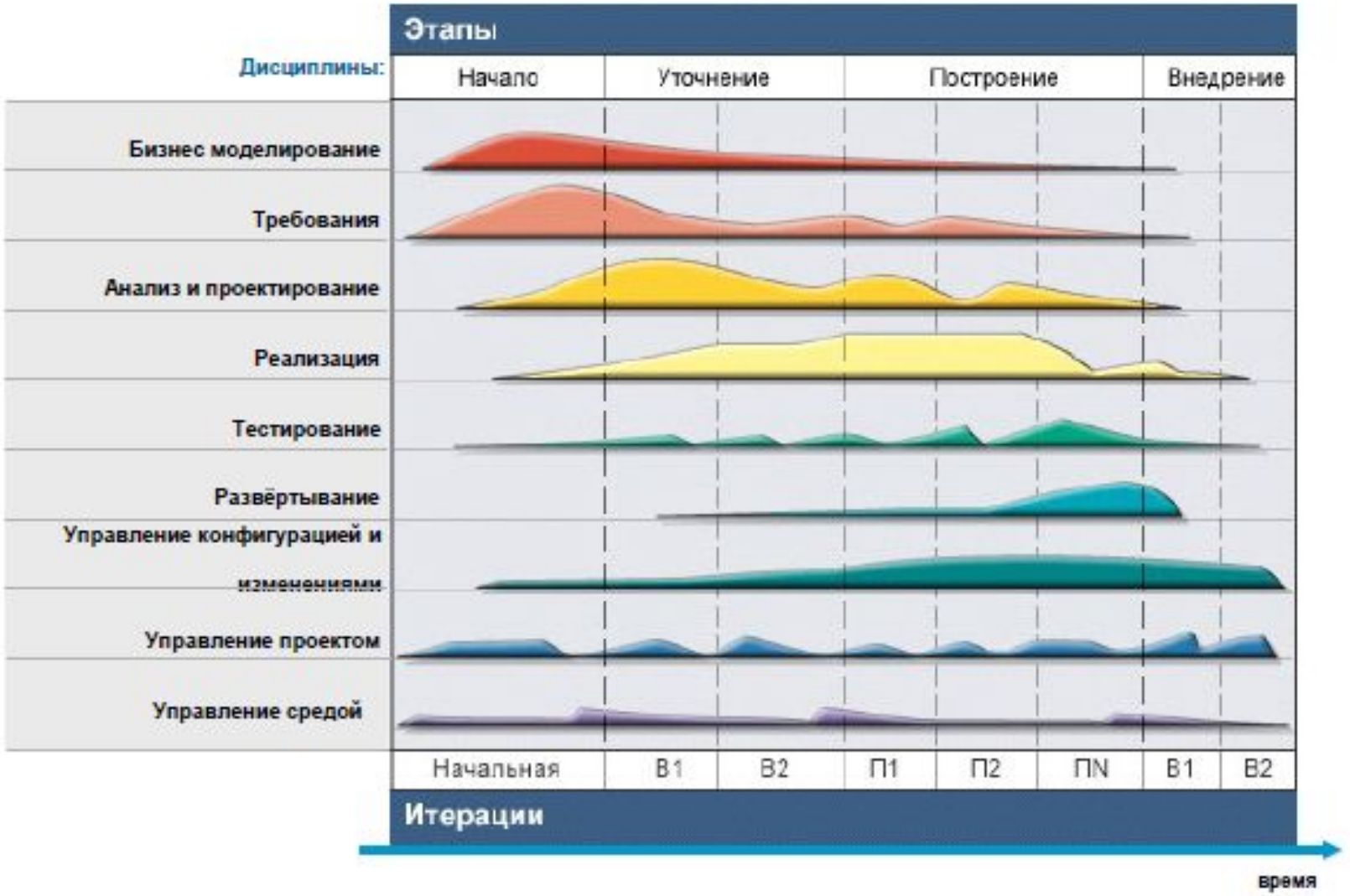
Шесть лучших практик RUP (после 2005 г.)

- **Практика 5. Увеличивайте уровень абстракции–** объектно-ориентированная реализация. Компонентная технология: компонент – набор логически сгруппированных классов (задание интерфейсов – оперции+атрибуты):
  - Определение границ между компонентом и пользователями компонента
  - Высокий уровень абстракции
  - Повторное использование компонентов
  - Независимость поддержки компонентов
  - Поддержка работы рассредоточенных команд
- **Практика 6. Постоянно уделяйте внимание качеству** – во всех дисциплинах (процессах) создания ИС
  - Дисциплина «Руководство проектом» - Планирование итерации включает формирование списка рисков, требований для реализации и внесения изменений; оценка результата итерации; анализ использования всех ресурсов
  - Дисциплина «Требования» – проверка качества документирования требований: соответствие реальным потребностям; доказуемость требований; понятность и доступность описания
  - Дисциплина «Анализ и проектирование» – проверка соответствия результатов проектирования (артефактов) требованиям; согласование уровня детализации; возможность преобразования проектных решений в исполняемый код
  - Дисциплина «Тестирование» - Тестирование в каждой итерации ближе к концу.

2. Архитектура технологии RUP

- Два ортогональных измерения
- Горизонтальное измерение представляет временное или динамическое измерение процесса:
  - Жизненный цикл, стадии (phases), итерации, контрольные точки (milestones), как процесс разработки развертывается во времени
- Вертикальное измерение: статическая структура
  - Роли исполнителей (roles), Задачи (activity, действия, работы), результаты действий (artifacts), рабочие процессы (workflows) логически группируются в дисциплины (disciplines)

Архитектура RUP



Итерационность технологии объектно-ориентированного проектирования RUP (Rational Unified Process)

- Итерационность технологии - последовательность работ в рамках утвержденного плана, приводящая к созданию работоспособного варианта ПО (релиза)
- Итерация - реализация одного или нескольких функциональных требований (вариантов использования)
- Каждая фаза содержит одну или более итераций, которые направлены на создание промежуточных компонентов системы, необходимых для достижения бизнес-целей данной фазы.
- Каждая итерация связана с вариантом использования и включает полный цикл: анализ, проектирование, кодирование, тестирование и интеграцию

1). Начальная стадия (inception)

Цели:

- Понять границы проекта, установить ключевые высокоуровневые требования к системе
- Разработать экономическое обоснование (business case), его концепцию (vision), список рисков
- Добиться соглашения между заинтересованными сторонами для дальнейшего продвижения

Веха целей жизненного цикла (LCO – Lifecycle Objective)

Результаты стадии:

- Общее описание системы (основные требования)
- Начальная диаграмма вариантов (прецедентов использования) – модель бизнес-процессов
- Начальный проектный глоссарий
- Начальный бизнес-план
- План проекта, отражающий стадии и итерации
- Один прототип или несколько

Итерации могут отсутствовать на этой фазе, если все понятно, или делаются итерации на «выброс», демонстрация возможностей

2). Стадия Уточнения (проектирования) - Elaboration

Цели:

- Свести к минимуму технические риски
- Создать базовую архитектуру (спроектировать, реализовать и протестировать ключевые компоненты, интерфейсы и архитектурные механизмы) функциональности
- Понять затраты построения системы

Веха архитектуры жизненного цикла (LCA – Lifecycle Architecture)

Результаты

- Детальные требования к ПО в виде диаграммы вариантов использования – 80 %
- Перечень нефункциональных требований
- Описание **базовой архитектуры**:
  - Модель предметной области (классов объектов)
  - Технологическая платформа, определяющая основные элементы технологии реализации и их взаимодействия
- Работающий прототип (20 % вариантов использования)
- Уточненный бизнес-план (затраты, риски)
- План разработки всего проекта, отражающий итерации и критерии для каждой итерации (детально следующая итерация)

3). Стадия конструирования (реализации) - Construction

Цели:

- Построить первую работающую версию продукта (несколько внутренних и альфа-релизов, выпуск бета-версии со средствами инсталляции, справочной документации, учебного материала)

Веха начальной функциональной готовности (IOC – Initial Operational Capability)

- Результаты:
  - ПО, интегрированное на требуемых платформах (бета-версия)
  - Руководства пользователей
  - Описание текущей реализации
- План развертывания

4). Стадия внедрения (ввода в действие) - Transition

Цели:

- Создать окончательную версию продукта и отправить заказчику (тестирование продукта, настройка на эксплуатацию)

Веха готового продукта (PR – Product Release)

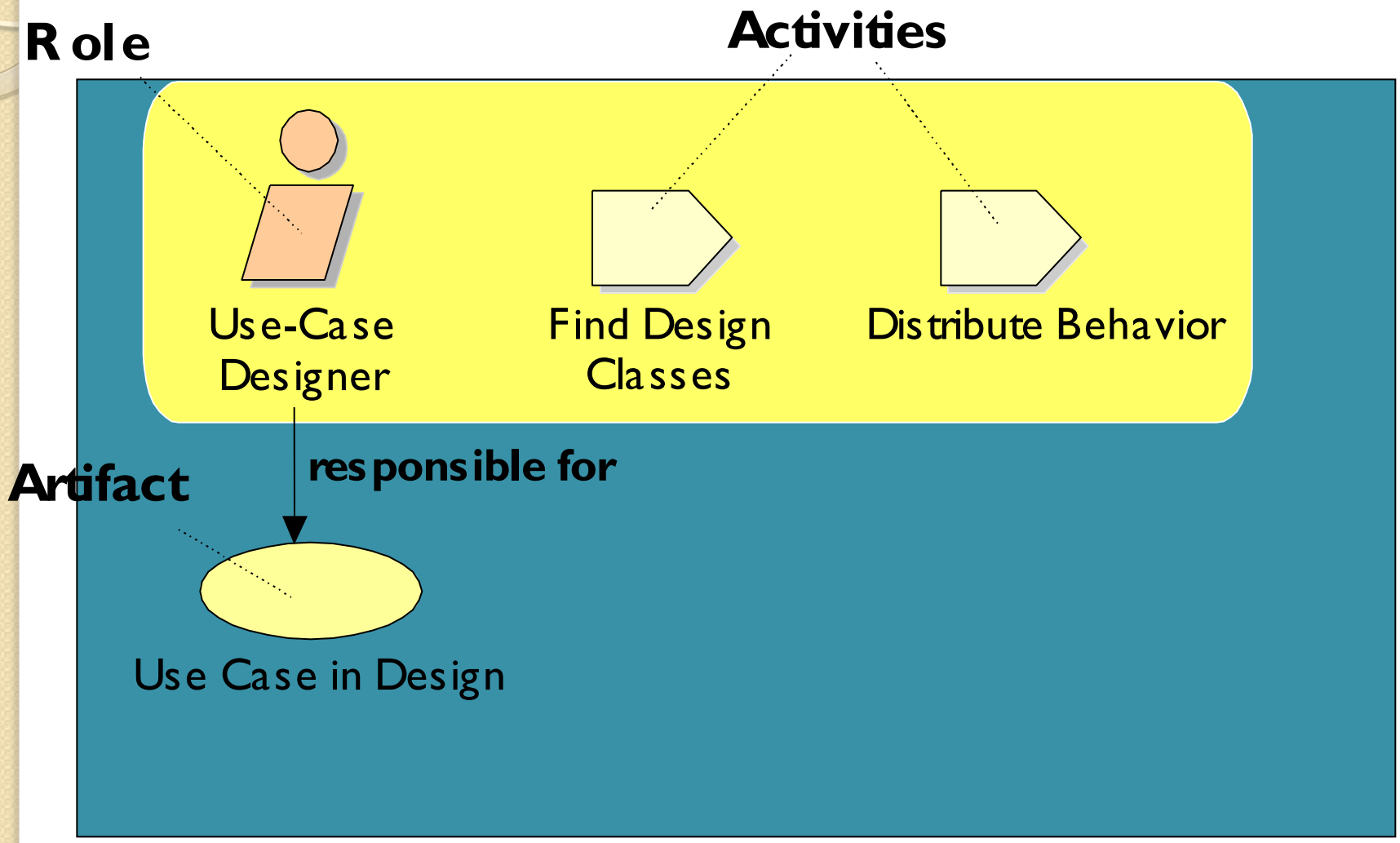
Результаты:

- Бета тестирование
- Параллельное функционирование с существующей системой
- Конвертирование баз данных
- Оптимизация производительности
- Обучение пользователей и специалистов ИТ служб

3. Статическая структура RUP

- Роли (Role) – категория исполнителей в процессе разработки рабочего продукта
- Действие (Activity – деятельность, задача) исполнителей нацелены на получение результата (рабочего продукта)
- Результат (Artifact) – конкретный артефакт (рабочий продукт): модель, документ, план, код и т.д.
- Рабочий процесс
  - Основные процессы: построение бизнес-моделей, определение требований, анализ и проектирование, тестирование, развертывание
  - Поддерживающие процессы: управление конфигурацией и изменениями, управление проектом, управление инфраструктурой (средой)

Роли, действия, рабочие продукты



Роль

- Роль указывает области компетенции и ответственности, которые должен иметь человек или группа людей при выполнении той или иной работы
- Роль определяет поведение и ответственность любого конкретного исполнителя или команды
- Поведение (полномочия): набор взаимосвязанных действий
- Ответственность: определяется по отношению к конкретным рабочим продуктам

Роли организации-подрядчика

- Руководитель проекта – отвечает за разработку и модификацию плана создания ИС (разработки ПО), а также его исполнение
- Технический лидер группы – старший разработчик с менеджерскими функциями
- Системный аналитик (требований)– отвечает за определение требований к ИС (ПО)
- Системный архитектор (ПО) – координирует решение технических задач и разработку артефактов во всем проекте, а также координирует принятие ключевых проектных решений, касающихся технологий, структуры и организации программной системы. Координация работы разных команд разработчиков в ее техническом аспекте, обеспечивает создание интерфейсов и контролирует их исполнение. Не несет ответственность за все проектные решения, а только за принципиальные.
- Разработчик (проектировщик) – проектирование, реализация и тестирование вариантов использования и компонентов (баз данных, интерфейсов с другими приложениями)
- Тестировщик – объективная оценка программного продукта на основе определенных критериев, таких как воспринимаемое качество, соответствие стандартам и выявление дефектов.
- Инженер управления конфигураций – ответственный за сборку и интеграцию программного продукта, внесение изменений, выпуск релизов (версий)
- Специалист по инструментальным средствам – поддержка среды проектирования

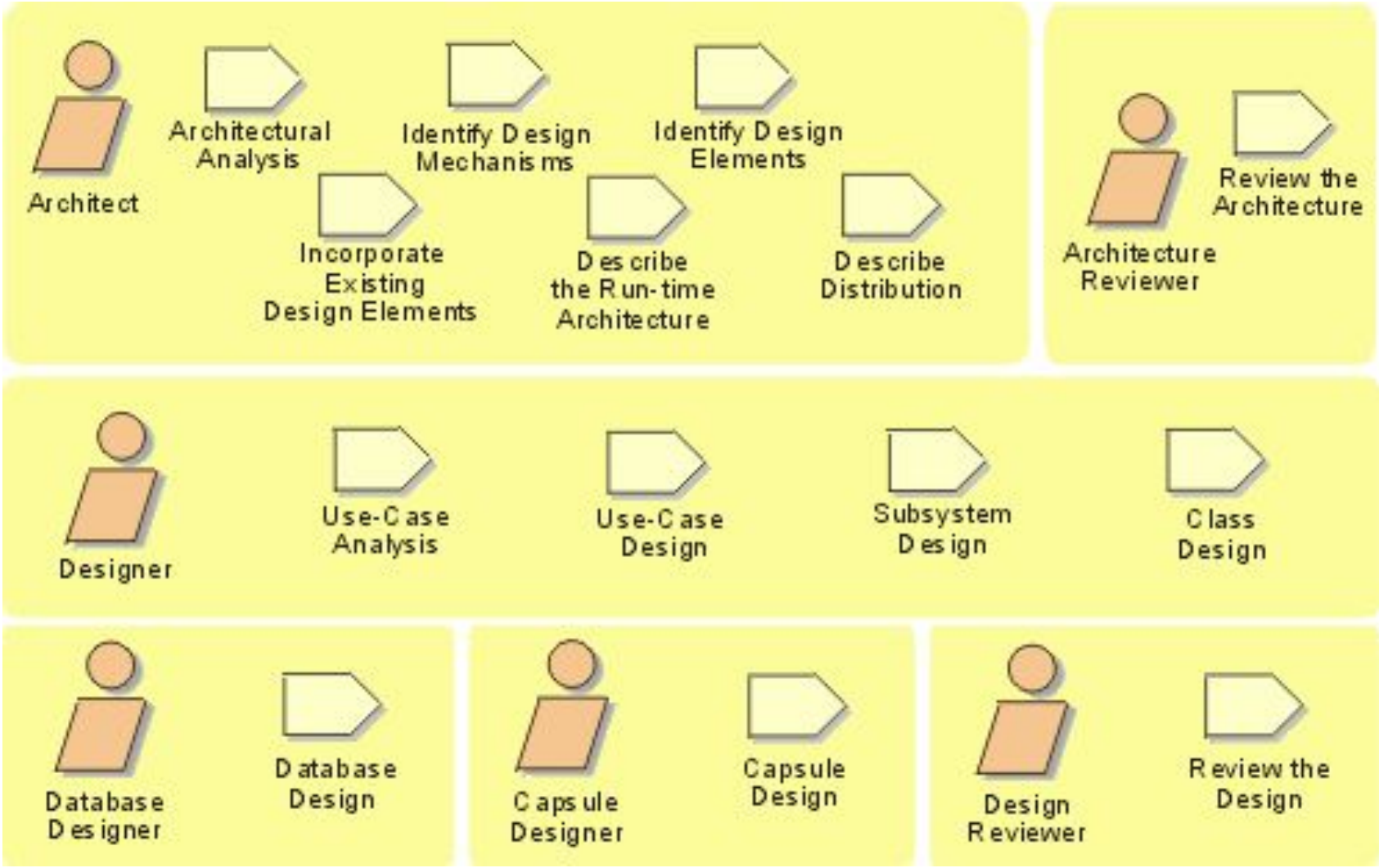
Роли организации-заказчика

- Руководитель проекта от заказчика – отвечает за планирование и исполнение проекта, управление ресурсами организации, выделенными на проект
- Внутренний лидер проекта – ведущий ключевой пользователь
- Архитектор проекта от заказчика – соблюдение стандартов и принципов ИТ-стратегии на предприятии
- Руководитель ИТ-службы – поддержа программно-технической среды на объекте автоматизации
- Ключевые пользователи – описание бизнес-процессов, формулирование бизнес-требований к системе.
- Специалисты по заключению контрактов – планово-экономическая работа

Действие (Activity, деятельность, задача)

- Единица работы исполнителя роли (например, создание или обновление артефактов)
- Степень детализации: требует исполнения от нескольких часов до нескольких дней
- Элемент планирования и оценки процесса
- Повторяется, при необходимости, в каждой итерации процесса

Пример: анализ и проектирование



Примеры действий

- Планирование итерации
- Определение вариантов использования и действующих лиц (Find use cases and actors)
- Обзор проектных решений (Review the design)
- Выполнение нагрузочного теста (Execute performance test)

Шаги (Steps) действий

- Действия разбиваются на шаги
- Виды шагов
  - Обдумывание (Thinking)
  - Выполнение (Performing)
  - Обзор результатов, проверка (Reviewing) и др.

Пример действия и шагов по созданию вариантов использования

Действие: Определение вариантов использования и действующих лиц

- Шаг: Определение действующих лиц
- Шаг: Определение вариантов использования
- Шаг: Описание взаимодействия вариантов использования и действующих лиц
- Шаг: Объединение вариантов использования и действующих лиц в пакеты
- Шаг: Построение диаграммы вариантов использования
- Шаг: Обзор модели вариантов использования
- Шаг: Оценка полученных результатов

Рабочий продукт (Artifact)

- Объект (объем информации), создаваемый, модифицируемый или используемый в некотором процессе
- Определяет область ответственности исполнителя и является результатом выполнения действий
- Виды рабочих продуктов:
  - Модели (напр., Use-Case Model) и элементы модели (напр., Use-Case)
  - Документы (напр., планы)
  - Исходный программный код
  - Исполняемые файлы (напр., прототип)
- Рабочие продукты могут содержать другие рабочие продукты

Примеры рабочих продуктов

- Проектная модель (Design model)
- Класс
- Вариант использования
- Тест
- План разработки ПО
- Релиз
- Оценка состояния проекта
- Перечень рисков

Рабочий процесс (Workflow)

- Последовательность действий, направленных на получение значимого результата (рабочего продукта) и описывающий взаимодействие категорий исполнителей (ролей)
- Основные процессы:
  - Построение бизнес-моделей (Business Modelling)
  - Управление требованиями (Requirements Managing)
  - Анализ и проектирование (Analysis and Design)
  - Реализация (Implementation)
  - Тестирование (Test)
  - Развертывание (Deployment)
- Поддерживающие процессы:
  - Управление конфигурацией и изменениями (Change Management)
  - Управление проектом (Project Management)
  - Управление инфраструктурой (Environment)

Основные процессы

- Business modeling (моделирование бизнес-процессов) - предполагает анализ бизнес-целей и бизнес-процессов, определение желаемых параметров системы и потребностей пользователей
- Requirements (требования) - предполагает сбор требований и их анализ, управление требованиями, перевод требований в функциональные спецификации. Здесь начинается анализ прецедентов и построение use cases (вариантов использования) - формальное отображение требований пользователя в UML;
- Analysis and design (анализ и проектирование) - предполагает перевод собранных требований в формализованную модель проекта. Результатом является описание системы на фазе проектирования и реализации - это документы уровня разработчиков системы. Язык формализации - Unified Modelling Language (UML). В процессе итеративной разработки выполнится эволюция: модель анализа -> модель проекта.
- Implementation (реализация, кодирование) - предполагает собственно написание кода. Элементы кода в RUP уже созданы на этапе анализа и проектирования, так как средство реализации UML - Rational Software Architect - позволяет создавать элементы кода на нескольких языках программирования. Методология - объектно-ориентированное программирование;
- Test (тестирование) - предполагает тестирование продукта на каждой итерации. Стоит специально отметить, что regression testing (возвратное тестирование) в данном случае должно содержать все актуальные тесты от предыдущей итерации и новые приемосдаточные тесты;
- Deployment (внедрение) - предполагает установку продукта на полигоне заказчика, подготовку персонала, запуск системы плюс приемо-сдаточные испытания, подготовка стандартов развертывания продукта, передача материалов отделу продаж (действия опциональны в зависимости от специфики продукта).

Поддерживающие процессы

- Configuration management (управление конфигурацией и изменениями) - процессы, направленные на управление версиями продукта, что предполагает контроль исходного кода (модели, исполняемых модулей, тестов, документации), контроль версий продукта, корпоративные стандарты разработки кода и документации, отслеживание изменений и ошибок (bug tracking); тесно связан с тестированием и поддержкой пользователей (customers support);
- Project Management (управление проектом) - предполагает набор административных действий управления проектом согласно идеологии RUP, используются средства управления проектом Rational;
- Environment (управление средой проектирования) - предполагает создание и поддержку средств анализа, проектирования, разработки, тестирования.

Дополнительные ресурсы RUP

- Основные понятия (Concepts)
  - Вводят основные определения, ключевые идеи, принципы
- Руководящие указания (Guidelines)
  - Методики, правила, эвристики, контрольные таблицы для выполнения действий, шагов, и обращения к артефактам
- Руководства по использованию инструментальных средств (Tool mentors)
  - Практическое руководство к использованию средств разработки
- Шаблоны (Templates)
  - для основных рабочих продуктов (артефактов)

Дисциплины RUP

- Все элементы процесса разработки – роли, действия, артефакты и связанные с ними дополнительные ресурсы сгруппированы в логические контейнеры, называемые дисциплинами, и соответствуют девяти основным и вспомогательным процессам разработки

Инструментальный комплекс RUP – база знаний и руководство

- Руководство для всех участников проектной команды
- Наставления по использованию инструментальных средств Rational Software Architect
- Примеры и шаблоны проектных решений
- Шаблоны проектной документации
- Шаблоны в формате MS Word
- Планы в формате MS Project

Руководящие указания

- Представляют собой правила, рекомендации, эвристики, поддерживающие действия и отдельные шаги
  - краткое, сжатое описание действий и шагов
  - правила качественного оформления результатов
  - конкретные методики
    - трансформация одного рабочего продукта в другой
    - использование UML
- Используются для оценки качества рабочих продуктов
- Адаптированы к условиям объекта внедрения технологии

Примеры руководящих указаний

- Руководства по моделированию:
  - классы, варианты использования, пакеты, модели
- Руководства по программированию
- Руководства по проектированию пользовательского интерфейса
- Контрольные точки
  - для оценки результатов

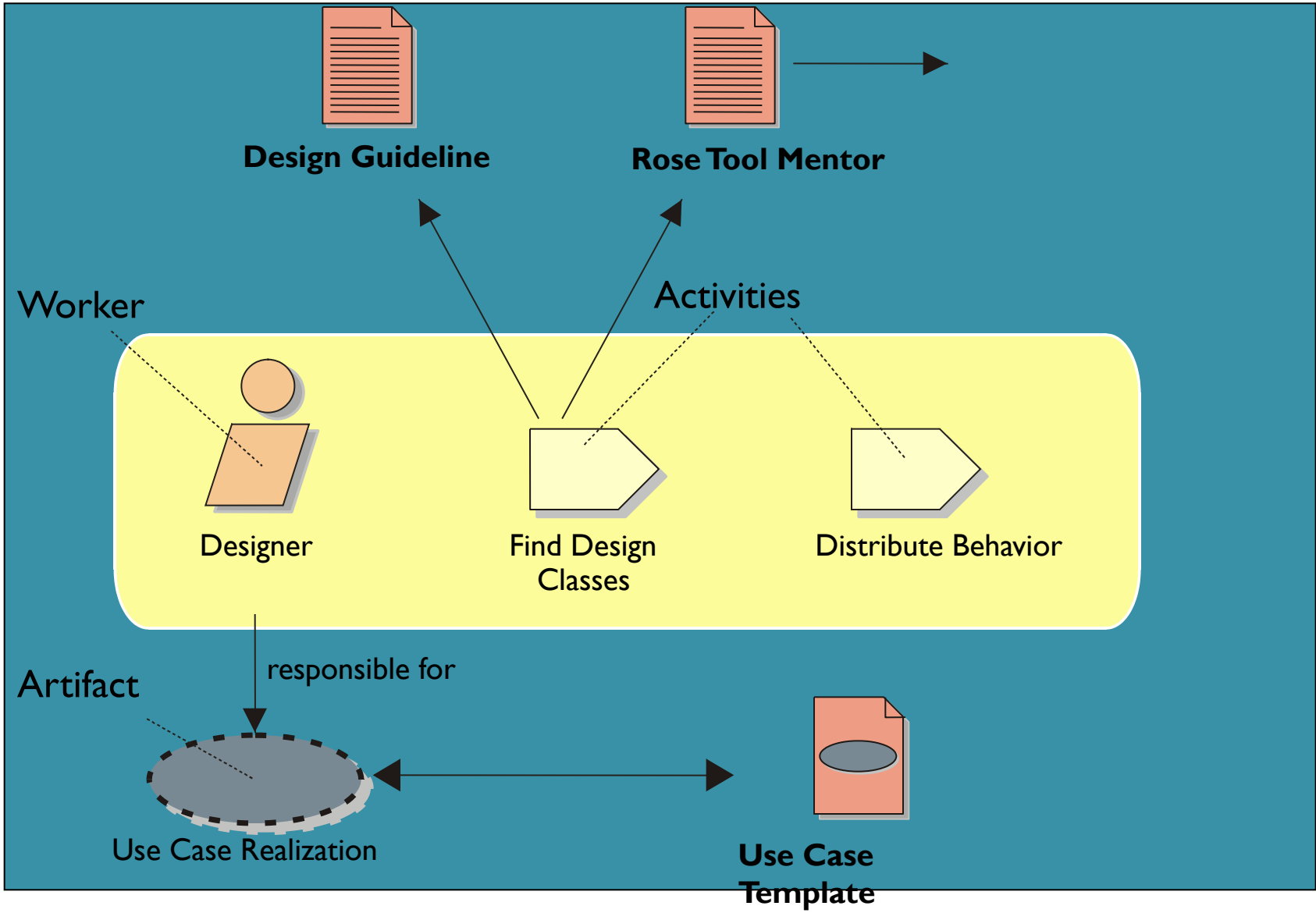
Руководства по использованию инструментальных средств

- Аналогичны руководящим указаниям
- Описывают использование конкретного инструментального средства для выполнения некоторых действий или их шагов
- Связаны со средствами Rational:
  - RequisitePro
  - Rational Software Architect
  - и др.

Шаблоны

- Предопределенные рабочие продукты, прототипы:
  - шаблоны Rational SoDA
  - шаблоны документов MS Word
  - планы MS Project
  - шаблоны страниц MS FrontPage
- Связаны с конкретными действиями, порождающими соответствующие рабочие продукты
- Адаптированы к условиям объекта внедрения технологии

Руководящие указания, руководства, шаблоны



Сравнение RUP с гибкими (Agile) технологиями проектирования

- Делать итерации как можно более короткими по времени – получение пригодной для демонстрации версий
- Организовать тесное взаимодействие всех участников проектной команды (аналитиков, архитекторов, разработчиков и тестировщиков) и с заказчиком
- Разработка приемочных тестов для отдельных функций и приемочного теста в конце
- Автоматизация тестирования, регрессионное тестирование + новый функционал
- Минимизация документирования, согласование получаемых артефактов (результатов) с заказчиком