Rational Unified Process (RUP)

Рациональный унифицированный процесс (Rational Unified Process, RUP)

- Корпорация **Rational Software** (www.rational.com)
  - формализовала технологический процесс разработки ПО
  - выпустила на рынок структурированную базу знаний под названием Rational Unified Process (**RUP**), в которую вошли
    - методические рекомендации ведущих разработчиков по эффективному созданию приложений и систем.
- RUP создана в виде страниц формата HTML:
  - обширная система гиперссылок,
  - графическая навигация,
  - подробное оглавление,
  - встроенный поисковый механизм.
- В качестве языка моделирования в общей базе знаний используется язык Unified Modelling Language (UML).

Rational Unified Process как технология

- **Rational Unified Process** - процесс разработки программного обеспечения.
- **Процесс** - частично упорядоченный набор шагов, которые нужно проделать для достижения цели.
- При разработке ПО **цель**: формирование или расширение существующего программного изделия.
- **Цель RUP**: гарантировать высокое качество программного продукта, отвечающего потребностям конечных пользователей, в пределах предсказуемого временного графика и бюджета.
- RUP обеспечивает строгий подход к назначению задач и ответственности в пределах группы разработки.

Характеристика RUP

Процесс:

- итеративный
- управляемый
- заключается в создании и обслуживании моделей
- сосредотачивает внимание на первоначальной разработке и компоновке устойчивой архитектуры программы
- поддерживает объектно-ориентированную технологию
- с перестраиваемой конфигурацией
- поддерживается инструментальными средствами, которые автоматизируют большинство действий процесса

Принципы RUP

- 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 (от 2 до 6 недель) проектная команда должна достичь
  - запланированных на данную итерацию целей,
  - создать или доработать проектные артефакты
  - получить промежуточную, но функциональную версию конечного продукта.
- Итеративная разработка позволяет
  - быстро реагировать на меняющиеся требования,
  - обнаруживать и устранять риски на ранних стадиях проекта,
  - эффективно контролировать качество создаваемого продукта.

Итерационный цикл

- Итерационный цикл основывается на
  - постоянном расширении
  - дополнении системы в процессе нескольких итераций с периодической обратной связью
  - адаптацией добавляемых модулей к существующему ядру системы.
- Система постоянно разрастается шаг за шагом, поэтому такой подход называют итерационным и инкрементным.
- Такой подход исключает
  - слишком быстрое написание кода (без детальной проработки
  - чрезмерно длительный этап детального проектирования
  - построения моделей без обратной связи.

Итерационный цикл

Последовательность нарастающих шагов или итераций.

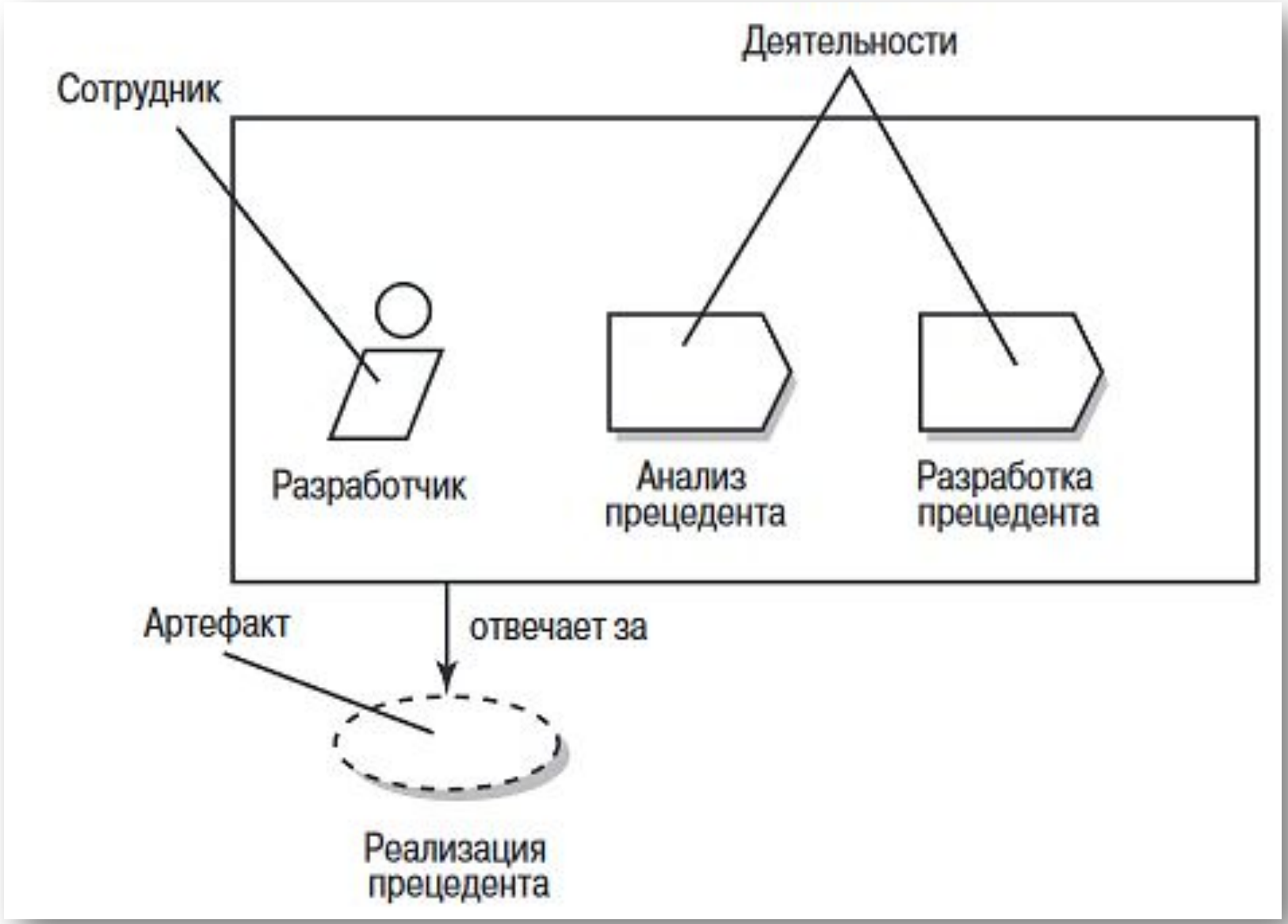
- Каждая итерация включает в себя некоторые или большую часть дисциплин разработки
  - выявление требований,
  - анализ,
  - проектирование,
  - реализация и т.п.
- У каждой итерации есть четко определенный набор целей, и она создает частично работающую реализацию конечной системы.
- Каждая последующая итерация строится на результатах предыдущих, развивает и усовершенствует систему до тех пор, пока не будет создан конечный продукт.
- Более ранние итерации больше концентрируются на требованиях, анализе и проектировании, более поздние - на реализации и тестировании.

Структура RUP

Процесс описывает, кто, что, как и когда делает:

- Сотрудники (workers) - “кто”
- Деятельности (activities) - “как”
- Артефакты (artifacts) - “что”
- Рабочие процессы (workflows) - “когда”

Структура RUP



Статический аспект RUP

Представлен четырьмя основными элементами:

- артефакты (рабочие продукты)
- роли
- виды деятельности
- дисциплины

- «**Роль**» (role) определяет
  - поведение
  - ответственность личности или группы личностей, составляющих проектную команду.

Одна личность может играть в проекте много различных ролей.

- Под **видом деятельности** конкретного исполнителя понимается единица выполняемой им работы (соответствует понятию технологической операции):
  - планирование итерации,
  - определение вариантов использования и действующих лиц
  - выполнение теста на производительность.

Каждый вид деятельности связан с конкретной ролью.

- **Артефакты** — это некоторые продукты проекта, порождаемые или используемые в нем при работе над окончательным продуктом:
  - модель, элемент модели,
  - документ,
  - исходный код, план

Дисциплина (discipline)

Дисциплина (discipline) соответствует понятию технологического процесса и представляет собой последовательность действий, приводящую к получения значимого результата.

В рамках RUP определены шесть основных дисциплин:

- построение бизнес - моделей
- определение требований
- анализ и проектирование
- реализация, кодирование
- тестирование
- развертывание, внедрение

и три вспомогательных:

- управление конфигурацией и изменениями
- управление проектом
- создание инфраструктуры.

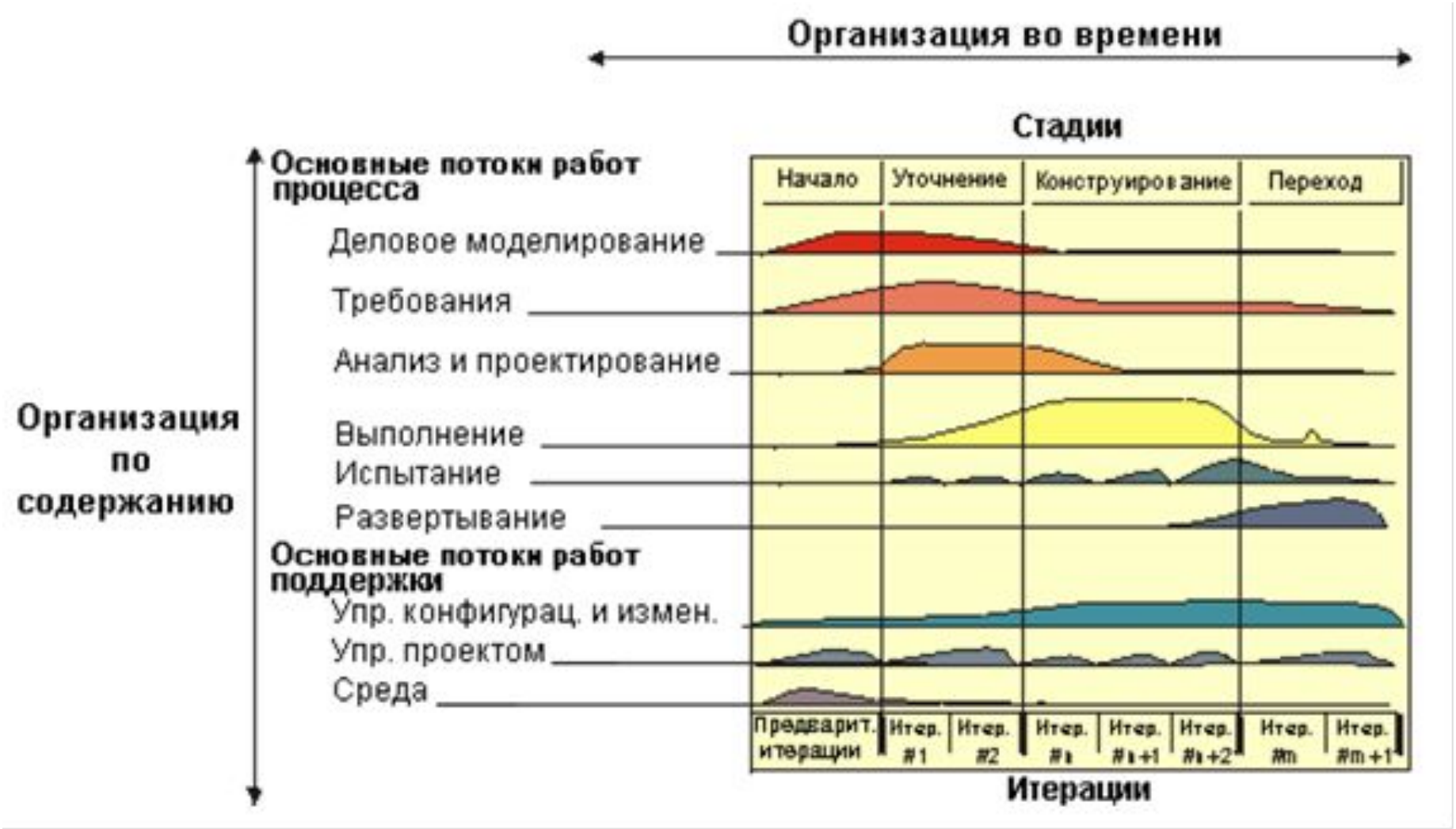
Жизненный цикл разработки

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

1. Начало (Inception)
2. Проектирование (Elaboration)
3. Построение (Construction)
4. Внедрение (Transition)

Итерационный цикл

Общее представление RUP



Общее представление RUP

Общее представление RUP в двух измерениях:

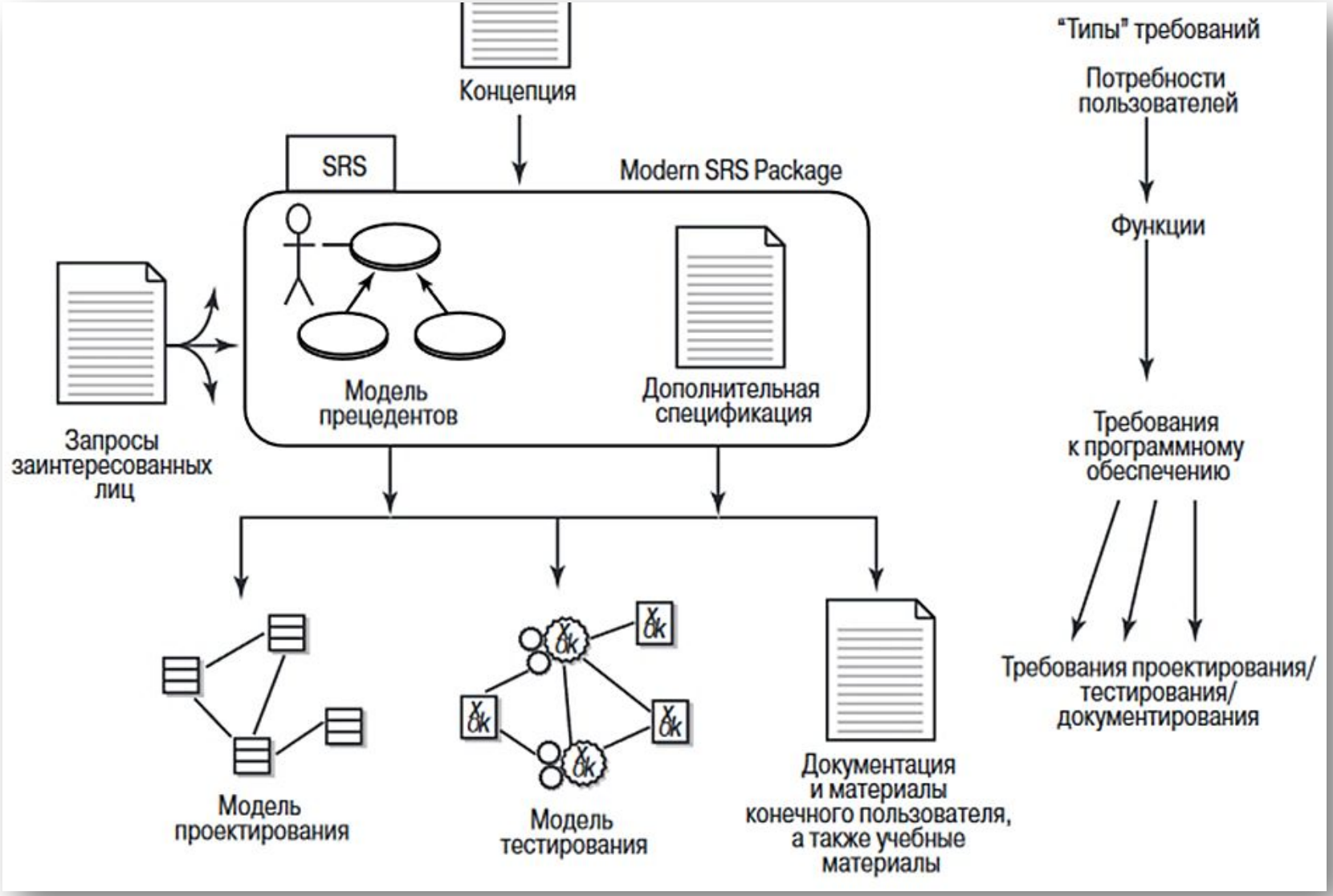
- горизонтальное измерение представляет
  - время
  - отражает динамические аспекты процессов
  - оперирует такими понятиями
    - стадии,
    - итерации
    - контрольные точки;
- вертикальное измерение отражает
  - статические аспекты процессов
  - оперирует такими понятиями
    - виды деятельности (технологические операции),
    - рабочие продукты,
    - исполнители
    - дисциплины (технологические процессы).

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.

Принципы управления требованиями в RUP



Артефакты рабочего процесса разработки требований

- **Запросы заинтересованных лиц** – коллекция различных запросов:
  - формальные запросы изменений,
  - потребности или другие пожелания заинтересованных лиц на протяжении жизненного цикла проекта, которые могут повлиять на требования к продукту
- **Документ-концепция** (Vision document):
  - кратко характеризует концепцию рассматриваемой системы: основные характеристики, функции, потребности заинтересованных лиц, а также основные предоставляемые услуги.

Артефакты рабочего процесса разработки требований

- **Модель прецедентов**, которая представляет собой организованный набор прецедентов, составляющих основную массу требований.
- **Дополнительные спецификации**, фиксирующие все требования, которые невозможно непосредственно связать с конкретными прецедентами:
  - нефункциональные требования и ограничения проектирования.
- Эти артефакты вместе образуют единую форму – пакет **Modern SRS Package**.

Modern Software Requirements Specification – Спецификация требований к программному обеспечению

- Представляет собой набор артефактов, полностью описывающих внешнее поведение системы.
- Создает концептуальную модель создаваемой системы.
- Исходная информация для создания Modern SRS Package – документ-концепция (Vision document), определяющий потребности пользователей, цели, задачи, целевые рынки и функции системы
- В пакете Modern SRS Package основное внимание уделяется деталям реализации этих функций.

Стандарт IEEE 830-1998

- Методика составления спецификаций требований к программному обеспечению, рекомендуемая Институтом Инженеров по Электротехнике и Радиоэлектронике (IEEE)
  - Описывается содержание и качественные характеристики правильно составленной спецификации требований к программному обеспечению (SRS) и приводится несколько образцовых SRS

Структура SRS

1. Введение
   - 1.1 Назначение
   - 1.2 Область действия
   - 1.3 Определения, акронимы и сокращения
   - 1.4 Публикации
   - 1.5 Краткий обзор
2. Полное описание
   - 2.1 Перспектива изделия
   - 2.2 Функции изделия
   - 2.3 Характеристики пользователя
   - 2.4 Ограничения
   - 2.5 Допущения и зависимости
3. Специфические требования

Структура SRS

3. Специфические требования

В приложении стандарта IEEE 830-1998 приводятся шаблоны раздела 3:

- По режимам
- По классам пользователей
- По объектам
- По свойствам
- По стимулам
- По функциональной иерархии
- Показывающий множественную организацию

Уточнение (Elaboration)

На этапе **Уточнения (фаза развития)** производится анализ предметной области и построение исполняемой архитектуры.

Фаза развития — это первая последовательность итераций, в течение которых решаются следующие задачи:

- Изучается и стабилизируется большая часть требований
- Обосновываются и устраняются основные риски
- Реализуются и тестируются базовые архитектурные элементы

Уточнение (Elaboration)

На начальных итерациях разрабатывается и обосновывается базовая архитектура:

- Разработка и реализация системы “не вглубь, а вширь”
  - означает идентификацию отдельных процессов, слоев, пакетов и подсистем, их высокоуровневых функций и интерфейсов, модули могут содержать в основном заглушки.
- Уточнение локальных и удаленных интерфейсов между модулями (включая параметры и возвращаемые значения).
- Интегрирование существующих компонентов
- Реализация упрощенных сценариев, обеспечивающих параллельное проектирование, программирование и тестирование основных компонентов

Планирование следующей итерации

- Требования и итерации систематизируются в соответствии с рисками, границами и критичностью
  - эти критерии используются для распределения работы по итерациям.
- **Риск** (risk) — это техническая сложность или другой фактор, например, отсутствие информации о необходимых затратах или ресурсах.
- **Границы** (coverage) — на начальных итерациях нужно определить все основные части системы, т.е. выполнить реализацию множества компонентов “не вглубь, а вширь”.
- **Критичность** (criticality) — требуется реализовать функции, имеющие важное значение для системы

Планирование следующей итерации

- Прецеденты или их отдельные сценарии ранжируются с целью определения приоритетов при реализации
- На начальных итерациях реализуются прецеденты с высоким рейтингом
- Ранжирование выполняется перед началом первой итерации, затем перед началом второй и т.д.
  - Такой план является адаптивным, а не зафиксированным на начальной стадии проекта

Приоритет	Требование (прецедент или свойство)	Комментарий
Высокий	Оформление продажи	Самый высокий приоритет
	Регистрация	Сложно добавить позднее
Средний	Поддержка пользователей	Влияет на безопасность

Уточнение (Elaboration)

Анализ предметной области и построение исполняемой архитектуры 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 означает достижение вехи архитектуры жизненного цикла (Lifecycle Architecture Milestone).

Построение (Construction)

Во время этой фазы происходит реализация большей части функциональности продукта.

Фаза **Построение** завершается

- первым внешним релизом системы
- руководством пользователя
- описанием текущей реализации
- вехой начальной функциональной готовности (Initial Operational Capability).

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,
- 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.

Business modeling (бизнес-анализ)

- **Артефакты-модели**:
  - модель бизнес-процессов - определение бизнес-требований к разрабатываемой системе;
  - модель структуры предприятия - артефакт для разработки функциональной модели системы;
  - модели документов, бизнес-сущностей, модели сценариев бизнес-функций, модели состояний бизнес-сущностей - для проектирования пользовательского интерфейса, БД системы; представляют собой описание статического и динамического состояний системы с различных точек зрения;
  - модели бизнес-правил - артефакт используется для моделирования правил в ПО.
- **Артефакты-документы**:
  - оценка организации заказчика, структура бизнеса;
  - словарь терминов предметной области;
  - набор бизнес-правил;
  - коммерческое предложение;
  - спецификации бизнес-функций;
  - план работ на этапе бизнес-моделирования;
  - рекомендации по проведению бизнес-моделирования;
  - запросы на изменение.

Requirements (требования)

- **Артефакты-модели**:
  - модель функции системы;
  - модель сценариев функций системы;
  - модель интерфейсов пользователя;
  - модель сценариев работы пользователя системы;
  - модель выходных форм;
  - модель правил системы.
- **Артефакты-документы**:
  - план управления требованиями;
  - словарь терминов системы;
  - спецификация на программную систему;
  - спецификация на функции системы;
  - правила системы;
  - запросы заинтересованных лиц;
  - план работ на этапе определения требований к системе;
  - рекомендации по моделированию на этапе определения требований;
  - запросы на изменение.

Analysis and design (анализ и проектироание)

- **Артефакты-модели**:
  - логическая модель данных;
  - физическая модель данных;
  - модель спецификаций компонентов системы;
  - сценарии взаимодействия классов, реализующих компоненты системы.
- **Артефакты-документы**:
  - архитектура программного обеспечения;
  - спецификации программных компонентов;
  - рекомендации на этапе анализа и проектирования;
  - план работ на этапе анализа и проектирования;
  - запросы на изменение.

Implementation (реализация, кодирование)

- **Артефакты-модели**:
  - компонентная модель приложения.
- **Артефакты-код**:
  - элементы генерации кода, полученные в Rational Rose;
  - собственно код приложения;
  - документация.
- **Артефакты-документы**:
  - план сборки приложения;
  - план работ на этапе реализации.

Test (тестирование)

- **Артефакты-модели**:
  - модель тестовых примеров;
  - функциональная модель тестовой программы;
  - модель спецификации компонентов тестовой программы;
  - сценарии взаимодействия классов, реализующих взаимодействие компонентов тестовой программы.
- **Артефакты-документы**:
  - описание тестовых примеров;
  - план тестирования;
  - план работ на этапе тестирования;
  - запросы на изменение.

Deployment (внедрение)

- **Артефакты-модели**:
  - модель размещения - описание размещения компонентов по узлам обработки.
- **Артефакты-документы**:
  - обучающие материалы;
  - документы по инсталляции;
  - описание версий системы;
  - план внедрения

Разработка в RUP

- Значительная часть RUP связана с разработкой и эксплуатацией моделей разрабатываемой системы.
- Модели помогают понимать и очерчивать как проблему, так и ее решение.
- Модель - это упрощение действительности, помогающее охватить большую, сложную систему, не поддающуюся пониманию во все своей полноте.
- Основной упор в RUP делается не на подготовку документов как таковых, а на моделирование разрабатываемой системы.
- Модели помогают очерчивать как проблему, так и пути ее решения, и создаются они при помощи унифицированного языка Unified Modeling Language (UML), предложенного компанией Rational и впоследствии утвержденного OMG как стандарт

UML (Unified Modeling Language)

- Унифицированный язык моделирования UML (Unified Modeling Language) - это графический язык визуализации спецификации и документирования артефактов преимущественно программной системы.
- Язык UML
  - представляет собой стандартное средство создания чертежной системы
  - определяет конкретные понятия
    - классы, написанные на определенных языках программирования,
    - схемы баз данных
    - программные компоненты с возможностью повторного использования.
  - позволяет разработчикам определять, визуализировать, конструировать и документировать артефакты программных систем.