

ТЕМА 2.

Технологии

проектирования

информационных систем

Лекция 6.

Современные технологии
проектирования ИС

Современные технологии проектирования

Название	Сокращение	Разработчик
Rational Unified Process	RUP	IBM (Rational Software)
Custom Development Method	CDM	Oracle
Microsoft Solutions Framework	MSF	Microsoft

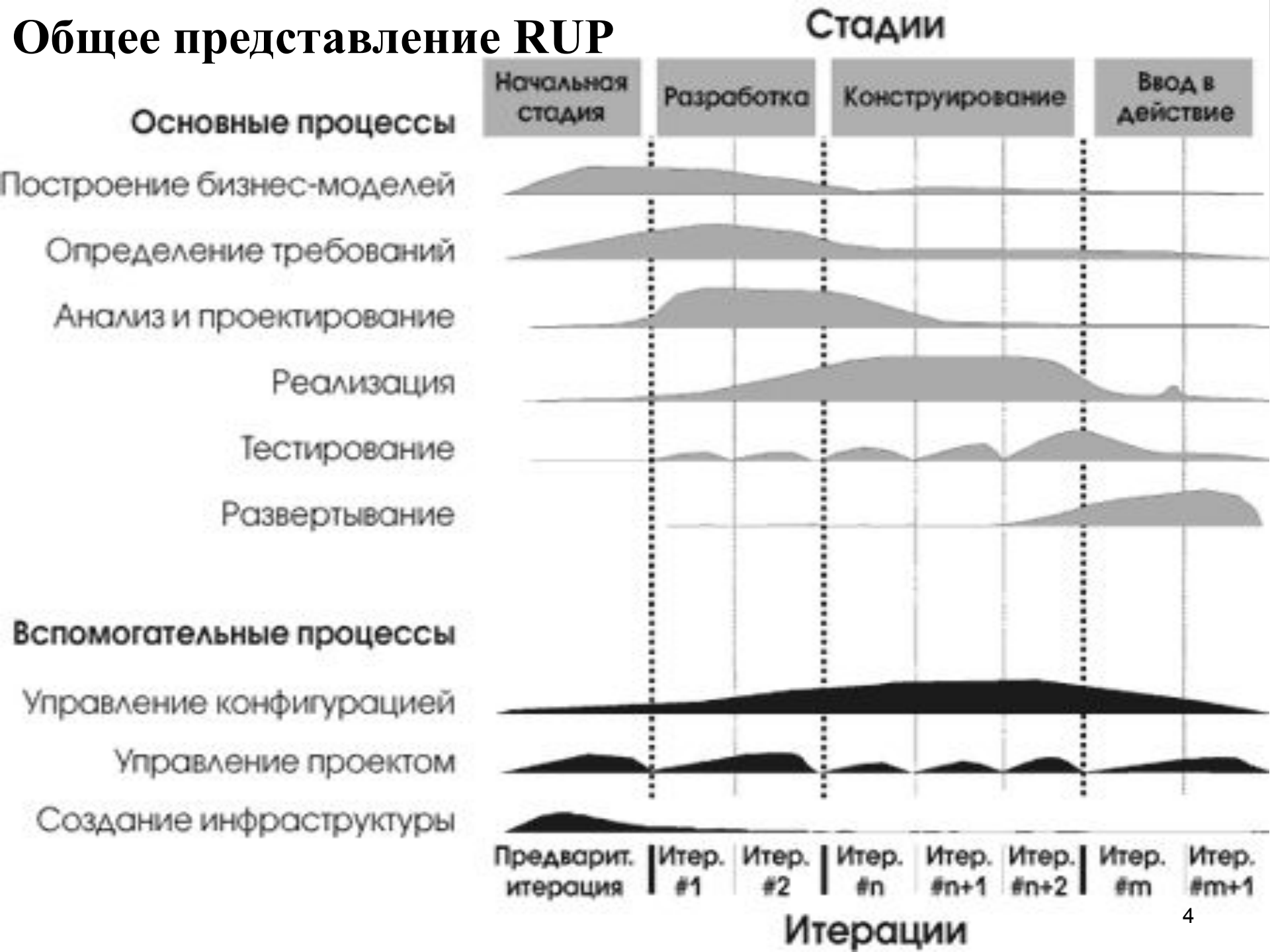
Технология Rational Unified Process

RUP соответствует стандартам и нормативным документам, связанным с процессами ЖЦ ПО и оценкой технологической зрелости организаций-разработчиков (ISO 12207, ISO 9000, CMM и др.).

Основные принципы:

- Итерационный и инкрементный (наращиваемый) подход к созданию ПО.
- Планирование и управление проектом осуществляется на основе функциональных требований к системе (вариантов использования).

Общее представление RUP



Начальная стадия RUP

Результаты:

- общее описание системы: основные требования к проекту, его характеристики и ограничения;
- начальная модель вариантов использования (степень готовности – 10-20%);
- начальный проектный глоссарий (словарь терминов);
- начальный бизнес-план;
- план проекта, отражающий стадии и итерации;
- один или несколько прототипов.

Стадия разработки RUP

Результаты:

- модель вариантов использования (завершенная на 80%), определяющая функциональные требования к системе;
- перечень дополнительных (нефункциональных) требований;
- описание базовой архитектуры будущей системы - модель предметной области и технологическую платформу;
- работающий прототип;
- уточненный бизнес-план;
- план разработки всего проекта, отражающий итерации и критерии оценки для каждой итерации.

Стадия конструирования RUP

- Стадия конструирования заключается в определении последовательности итераций конструирования вариантов использования, реализуемых на каждой итерации.
- **Результатом** стадии является продукт, готовый к передаче конечным пользователям:
 - ПО, интегрированное на требуемых платформах;
 - руководства пользователя;
 - описание текущей реализации.

Стадия ввода в действие

- предназначена для передачи готового продукта в распоряжение пользователей.
- Данная стадия включает:
 - бета-тестирование, позволяющее убедиться, что новая система соответствует ожиданиям пользователей;
 - параллельное функционирование с существующей системой, которая подлежит постепенной замене;
 - конвертирование баз данных;
 - оптимизацию производительности;
 - обучение пользователей и специалистов службы сопровождения.

Статический аспект RUP

1. **Роль** (role) – определяет поведение и ответственность личности (члена проектной команды).
2. **Вид деятельности** (activity) – единица выполняемой работы (технологическая операция), сопровождается набором руководств (guidelines), представляющих собой методики выполнения технологических операций.
3. **Рабочий продукт** (artifact) – модель, элемент модели, документ, исходный код или план, являющийся результатом вида деятельности.
4. **Дисциплина** (discipline) – последовательность действий, приводящая к получению значимого результата (технологический процесс).

Дисциплины RUP

Основные дисциплины:

- 1) построение бизнес-моделей;
- 2) определение требований;
- 3) анализ и проектирование;
- 4) реализация;
- 5) тестирование;
- 6) развертывание.

Вспомогательные дисциплины:

- 1) управление конфигурацией и изменениями;
- 2) управление проектом;
- 3) создание инфраструктуры.

Компоненты RUP

- Описание всех элементов динамического и статического аспекта RUP;
- навигатор по всем элементам RUP, глоссарий и средство быстрого обучения технологии;
- руководства для всех участников проектной команды, охватывающие весь жизненный цикл ПО;
- рекомендации по использованию инструментальных средств, входящих в состав Rational Suite;
- примеры и шаблоны проектных решений для Rational Rose;
- шаблоны проектной документации для SoDa;
- шаблоны в формате Microsoft Word, предназначенные для поддержки документации по всем процессам и действиям жизненного цикла ПО;
- планы в формате Microsoft Project, отражающие итерационный характер разработки ПО.

Инструментальные средства для поддержки RUP

RUP опирается на интегрированный комплекс инструментальных средств *Rational Suite*. Он существует в следующих вариантах:

- *Rational Suite AnalystStudio* – предназначен для определения и управления полным набором требований к разрабатываемой системе;
- *Rational Suite DevelopmentStudio* – предназначен для проектирования и реализации ПО;
- *Rational Suite TestStudio* – представляет собой набор продуктов, предназначенных для автоматического тестирования приложений;
- *Rational Suite Enterprise* – обеспечивает поддержку полного жизненного цикла ПО и предназначен как для менеджеров проекта, так и отдельных разработчиков, выполняющих несколько функциональных ролей в команде разработчиков.

Состав IBM Rational Suite

- *IBM Rational RequisitePro* – средство управления требованиями;
- *IBM Rational Rose* – средство визуального моделирования;
- *IBM Rational XDE* – средство генерации объектного кода;
- *IBM Rational RapidDeveloper* – средство разработки;
- *IBM Rational ClearCase* – средство конфигурационного управления;
- *IBM Rational ClearQuest* – средство управления изменениями;
- *IBM Rational SoDA* – средство автоматизированного документирования;
- *IBM Rational Quantify* – средство количественного определения узких мест, влияющих на общую эффективность работы программы;
- *IBM Rational TestManager* – средство планирования функционального и нагрузочного тестирования;
- *IBM Rational Robot* – средство записи и воспроизведения тестовых сценариев;
- *IBM Rational TestFactory* – средство тестирования надежности;
- *IBM Rational Quality Architect* – средство генерации кода для тестирования.

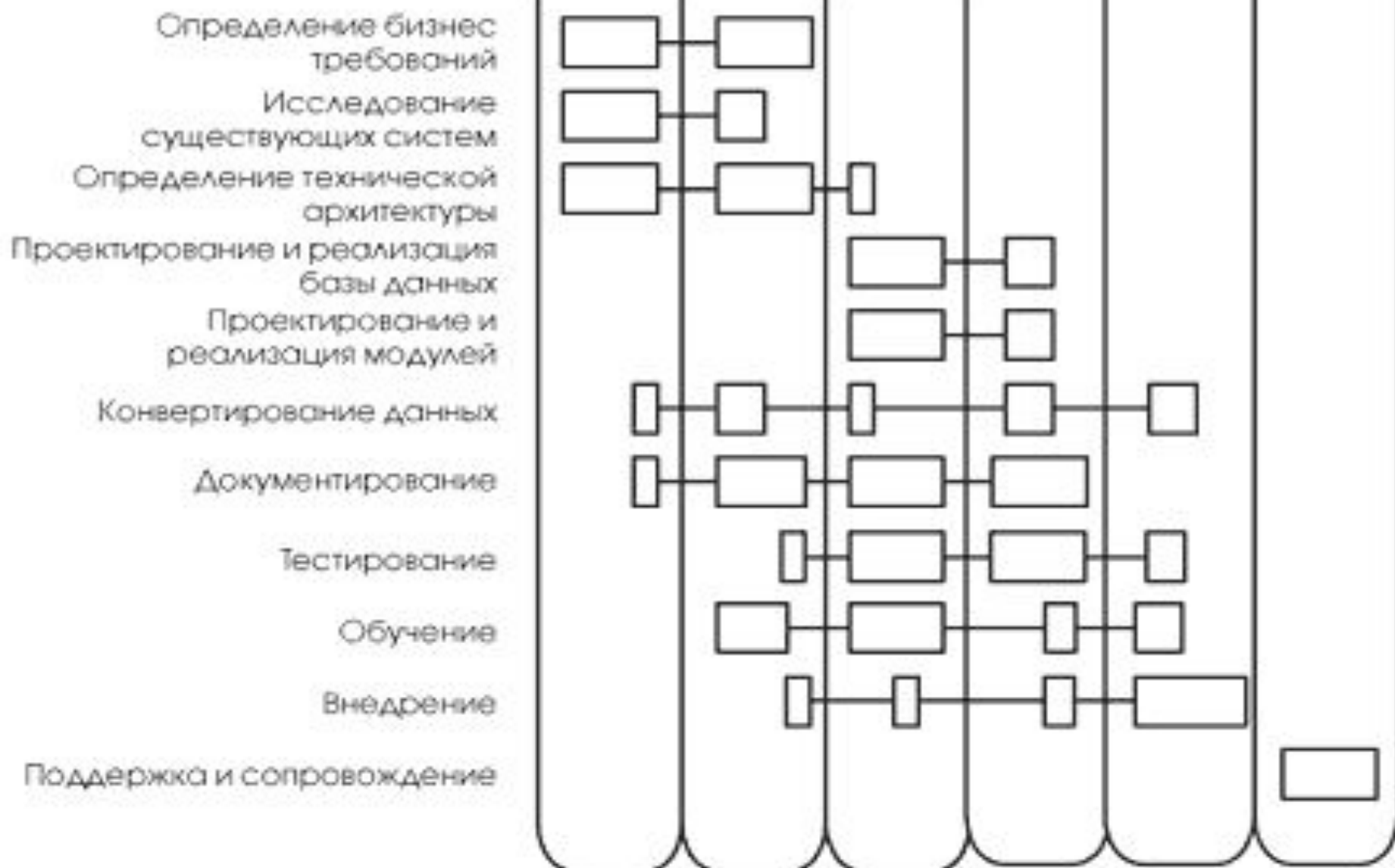
Технология Custom Development Method

- Методическая основа технологии создания ПО корпорации Oracle – комплекс методов, охватывающий большинство процессов ЖЦ ПО.
- В состав комплекса входят:
 - **CDM** (Custom Development Method) – разработка прикладного ПО;
 - **PJM** (Project Management Method) – управление проектом;
 - **AIM** (Application Implementation Method) – внедрение прикладного ПО;
 - **BPR** (Business Process Reengineering) – реинжиниринг бизнес-процессов;
 - **OCM** (Organizational Change Management) – управление изменениями.

Этапы CDM

Стратегия Анализ Проектирование Реализация Внедрение Эксплуатация

Процессы CDM



Стадии	Предназначение
Стратегия	Определение целей создания системы, приоритетов и ограничений, разработка системной архитектуры и формирование плана разработки.
Анализ	Построение модели информационных потребностей, диаграмм функциональной иерархии, матрицы перекрестных ссылок и диаграмм потоков данных.
Проектирование	Разработка подробной архитектуры системы, схемы реляционной БД и программных модулей, установление перекрестных ссылок между компонентами системы для анализа их взаимного влияния и контроля за изменениями.
Реализация	Создание БД, разработка и тестирование прикладных систем, проверка их качества и соответствия требованиям пользователей, разработка системной документации, материалов для обучения и руководства пользователей.
Внедрение	Анализ производительности и целостности системы.
Эксплуатация	Поддержка и модификация системы.

Критерии выбора метода разработки по CDM

При определении подхода к разработке оценивается:

- масштаб, степень сложности и критичность будущей системы;
- стабильность требований пользователей;
- сложность и количество бизнес-правил;
- количество автоматически выполняемых функций;
- разнообразие и количество пользователей;
- степень взаимодействия с другими системами.

Характеристики	Классический подход (каскадный)	Подход быстрой разработки (итерационный)
Количество этапов	5	4
Характеристики проекта	■ Высокая сложность ■ Большой масштаб ■ Нечетко определенная задача	■ Несложная архитектура системы ■ Небольшие и средние по масштабу проекты ■ Четкая постановка задачи
Характеристики исполнителей	Невысокая квалификация исполнителей, неподготовленные пользователи	Высококвалифицированные универсальные исполнители, хорошо подготовленные пользователи
Продолжительность проекта	8 – 36 месяцев	4 – 16 месяцев

Комплекс Oracle Developer Suite для быстрой разработки

- ***Oracle Designer*** - средство моделирования и генерации приложений;
- ***Oracle Forms*** - средство быстрой разработки приложений;
- ***Oracle Reports*** - визуальное средство разработки отчетов;
- ***Oracle JDeveloper*** - средство визуального программирования на языке Java;
- ***Oracle Discoverer*** - средство для разработки аналитических приложений;
- ***Oracle Warehouse Builder*** - система для построения хранилищ данных;
- ***Oracle Portal*** - средство разработки информационного портала организации.

Технология Microsoft Solution Framework

Microsoft Solutions Framework (MSF) - платформенно-независимая методология управления жизненным циклом приложений (application lifecycle management, ALM), представленная в виде согласованного набора концепций, моделей и правил.

Состав MSF:

- Модель процессов;
- Модель проектной группы;
- Дисциплина управления проектами;
- Дисциплина управления рисками;
- Дисциплина управления подготовкой.

Модель процессов MSF



Создание общей картины приложения

- Определение бизнес-целей;
- определение структуры проекта;
- **определение состава команды;**
- оценка существующей ситуации;
- **создание документа общей картины и области действия проекта;**
- определение требований и профилей пользователей;
- разработка концепции решения;
- оценка риска;
- закрытие этапа.

Планирование

Этап	Точка зрения	Результат
Концептуальное проектирование	Бизнес-требования, пользовательские требования	Набор сценариев использования системы
Логическое проектирование	Проектная команда	Набор сервисов
Физическое проектирование	Программисты	Набор используемых технологий и программных интерфейсов

Контрольные точки этапа планирования

- Функциональная спецификация;
- план управления рисками;
- определение среды разработки и тестирования;
- генеральный план и календарный график проекта.

Этап разработки

■ Задачи:

- создание прототипа приложения;
- разработка программных компонентов приложения;
- создание решения из подготовленных компонентов;
- закрытие разработки (реализация всех функций, готовность кода и документации).

■ Результаты:

- исходный текст кода и исполняемые файлы;
 - сценарии установки и конфигурации для развертывания;
 - окончательная функциональная спецификация;
 - спецификации и сценарии тестирования.
- **Контрольная точка – окончательное утверждение области действия проекта (все функции продукта готовы и прошли тестирование в рамках своего модуля).**

Стабилизация

■ Задачи:

- тестирование компонентов;
- тестирование баз данных;
- тестирование инфраструктуры;
- тестирование защиты;
- тестирование интеграции;
- анализ удобства работы с продуктом;
- нагрузочное тестирование (включая анализ ресурсоемкости и производительности);
- ведение отчетности по тестированию.

■ Результат:

подтверждение готовности продукта к выпуску и полноценному развертыванию в промышленной среде.

Развертывание

■ Задачи:

- установка решения и необходимых компонентов окружения;
- проведение стабилизации продукта в промышленных условиях;
- передача проекта группе сопровождения;
- анализ проекта в целом на предмет уровня удовлетворенности заказчика.

Модель проектной группы

- **Модель проектной группы MSF (*MSF Team Model*)** описывает подход *Microsoft* к организации работающего над проектом персонала и его деятельности в целях максимизации успешности проекта.
- **Модель проектной группы основана на:**
 - 6 принципах
 - 6 концепциях
 - 6 ролевых кластерах

Основные принципы модели проектной группы

1. Распределение ответственности при фиксации отчетности
2. Наделение членов команды полномочиями
3. Концентрация на бизнес-приоритетах
4. Единое видение проекта
5. Готовность к переменам
6. Свободное общение членов группы

Ключевые концепции модели проектной группы

1. Проектная группа – команда соратников
2. Сфокусированность на нуждах заказчика
3. Нацеленность на конечный результат
4. Установка на отсутствие дефектов
5. Стремление к самосовершенствованию
6. Заинтересованные команды работают эффективно

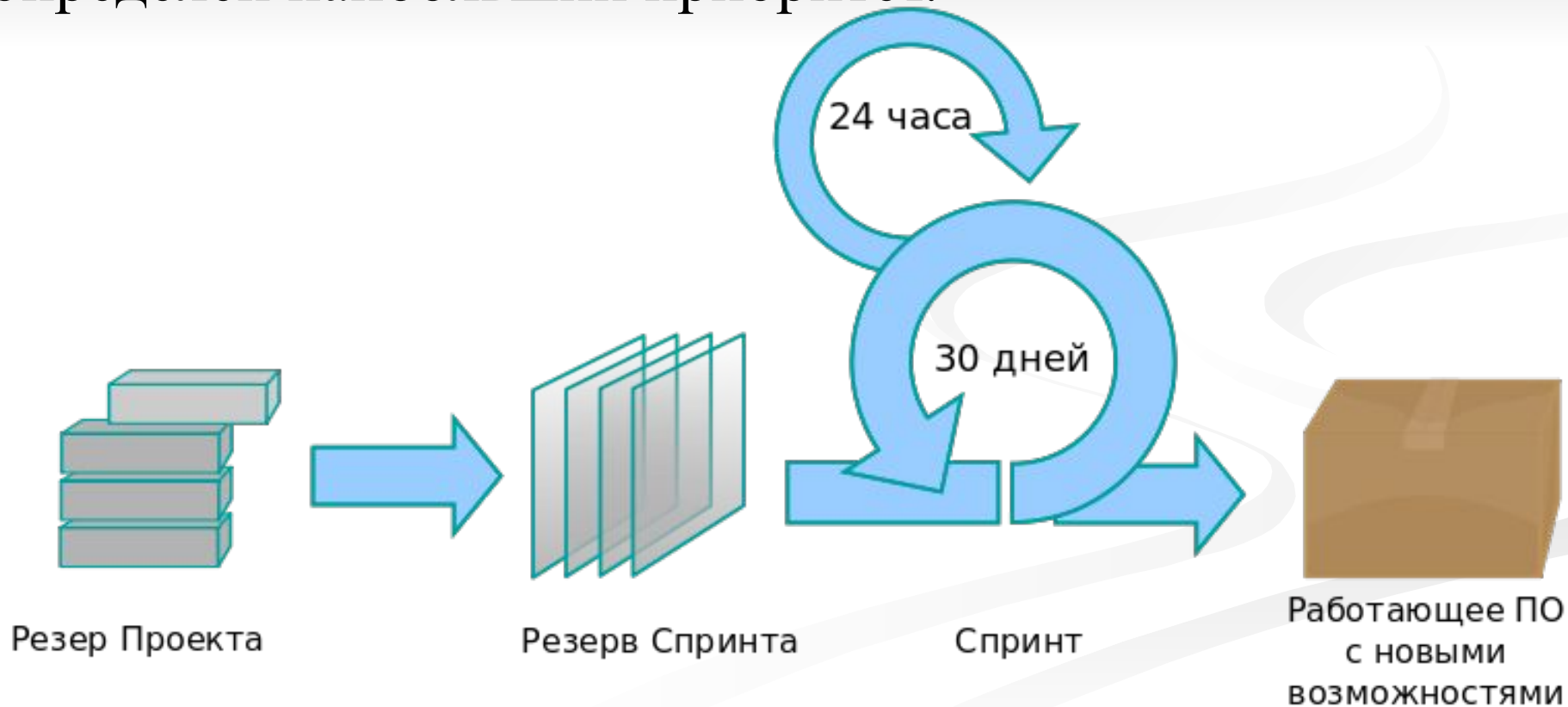
Ролевые кластеры

1. **Управление продуктом** (менеджер продукта - *product manager*) — бизнес-приоритеты, маркетинг, представительство интересов заказчика.
2. **Управление программой** (менеджер программы - *program manager*) — разработка архитектуры решения, административные службы
3. **Разработка** (разработчик - *developer*) — разработка приложений и инфраструктуры, технологические консультации
4. **Тестирование** (тестировщик - *tester*) — планирование, разработка тестов и отчетности по тестам
5. **Управление выпуском** (менеджер по выпуску- *release manager*) — инфраструктура, сопровождение, бизнес-процессы, выпуск готового продукта
6. **Удовлетворение заказчика** (специалист по удобству использования - *user experience*) — обучение, эргономика, графический дизайн, техническая поддержка

	<i>Менеджер продукта</i>	<i>Менеджер програм- мы</i>	<i>Разрабо- тчик</i>	<i>Тести- ровщик</i>	<i>Менеджер по выпуску</i>	<i>Спец. по удобству использо- вания</i>
<i>Менеджер продукта</i>		—	—	+	—/+	—/+
<i>Менеджер программы</i>	—		—	—/+	+	—/+
<i>Разработ- чик</i>	—	—		—	—	—
<i>Тестиров- щик</i>	+	—/+	—		+	+
<i>Менеджер по выпуску</i>	—/+	+	—	+		—/+
<i>Спец. по удобству использова- ния</i>	—/+	—/+	—	+	—/+	

Методология Scrum

позволяет в жёстко фиксированные и небольшие по времени итерации предоставлять пользователю работающее ПО с новыми возможностями, для которых определён наибольший приоритет.



Основные принципы Scrum

- **Люди и их взаимодействие** важнее процессов и инструментов;
- **Готовый продукт** важнее документации по нему;
- **Сотрудничество с заказчиком** важнее жестких контрактных ограничений;
- **Реакция на изменения** важнее следования плану.

Элементы Scrum

- **Спринт** — итерация (1-4 недели), в ходе которой обеспечивается функциональный рост ПО.
- **Резерв проекта** — список требований к функциональности, подлежащих реализации, упорядоченный по степени важности. Элементы списка называются «пожеланиями пользователя» (*user story*) или элементами резерва (*backlog items*). «Будучи пользователем <тип пользователя> я хочу сделать <действие>, чтобы получить <результат>».
- **Резерв спринта** — содержит функциональность, выбранную владельцем проекта из резерва проекта. Все функции разбиты по задачам, каждая из которых оценивается **скрам-командой**.

Основные роли Scrum

- **Скрам-мастер (ScrumMaster)** — проводит совещания (Scrum meetings) следит за соблюдением всех принципов скрам, разрешает противоречия и защищает команду от отвлекающих факторов.
- **Владелец продукта (Product Owner)** — представляет интересы конечных пользователей и других заинтересованных в продукте сторон, предоставляет понятные и тестируемые требования команде, отвечает за приемку кода в конце каждой итерации.
- **Скрам-команда (Scrum Team)** — команда разработчиков проекта, состоящая из специалистов разных профилей. Размер команды в идеале составляет 7 ± 2 человека.

Дополнительные роли Scrum

- Пользователи (*Users*)
- Клиенты, Продавцы (*Stakeholders*) — лица, которые инициируют проект и для кого проект будет приносить выгоду. Они вовлечены в скрам только во время **обзорного совещания по спринту**.
- Управляющие (*Managers*) — люди, которые управляют персоналом.
- Эксперты-консультанты (*Consulting Experts*)

Процессы Scrum

- Планирование спринта (4-8 ч.)
- Ежедневное совещание (15 мин.)
 1. Что было сделано с предыдущего совещания?
 2. Что будет сделано к следующему совещанию?
 3. Какие есть проблемы? (скрам-мастер)
- Скрам над скрамом (после ежедневного совещания в случае параллельной работы нескольких команд).
- Обзор итогов спринта (4 ч.).
- Ретроспективное совещание (1-3 ч.).

Традиционный подход	Agile
• Все требования должны быть определены и детально описаны до начала разработки; • Дорого и медленно; • Чувствителен к изменениям; • Мало возможностей для конечного пользователя повлиять на цели проекта и требования к продукту; • Зачастую проблемы выявляются на этапе тестирования; • Много документации, много технической документации, которая не понятна конечному пользователю или заказчику.	• Может привести к низкому качеству продукта; • Риск никогда не достигнуть закрытия/завершения проекта; • Могут возникнуть проблемы с расширяемостью продукта.

Традиционный подход	Agile
•Легок для понимания и использования; •Детально структурирован, что облегчает его применение к малоопытным командам; •Задаёт стабильные требования к проекту/продукту с самого старта; •Проекты легко контролируются, отслеживаются ресурсы, риски, время; •Качество имеет первоочередной приоритет по сравнению со стоимостью и временем.	•Итеративная разработка; •Использование временные рамки(time boxes); •Конечный пользователь вовлечен в процесс с самого начала; •Быстрое получение первой/пробной версии продукта для тестирования; •Легко воспринимаются корректировки и изменения в процессе разработки.

Традиционный подход	Agile
• Требования к продукту предельно ясны и стабильны; • Известны используемые технологии и инструменты; • Продукт четко формализован • Архитектура продукта строго регламентирована и детализована; • Требования внешних нормативных документов.	• Команда с высоким уровнем профессионализма; • Тесная связь заказчика и разработчиков; • вовлечен в проект со старта; • Четко определены бизнес-цели проекта/продукта; • Состав команды стабильный; • Технические требования приемлемые, коллериются с технологиями, которые собираются быть использованными для разработки; • Система может быть модульной.

Подходы к созданию ИС

1. **Разработка** (самостоятельно или силами другой компании)
 - Прототипирование
2. **Покупка** готового решения, его адаптация и настройка под специфику предприятия
 - Покупка ядра ИС и ее модификация
3. **Аренда** ИС у ASP провайдера (*Application Service Provider*).

Собственная разработка ИС

Достоинства	Недостатки
■ возможность разработки АИС для конкретных целей предприятия; ■ отсутствие функциональных, информационных и других ограничений, присущих готовым АИС; ■ повышение степени совместимости АИС с уже использующимися на предприятии системами.	■ большие затраты ресурсов; ■ сложность в определении пользователем своих потребностей; ■ необходимость в жестком планировании и контроле над разработкой; ■ необходимость адекватной оценки возможностей; ■ отсутствие необходимой квалификации у сотрудников.

Прототипирование

- ***Прототипирование*** – это подход к разработке ИС, при котором создается ее упрощенная действующая модель (прототип).
- **Условия использования:**
 - небольшая команда проектировщиков-универсалов (от 2 до 10 человек);
 - короткий, но тщательно проработанный производственный график (от 2 до 6 мес.);
 - использовании спиральной модели ЖЦ ИС;
 - тесное взаимодействие с заказчиком.

Прототипирование

Достоинства	Недостатки
■ лучшее определение потребностей пользователей; ■ большая вовлеченность пользователей в разработку; ■ ускорение времени разработки; ■ обнаружение многих ошибок при экспериментах; ■ простота внесения изменений; ■ меньшая стоимость	■ большой расход времени пользователей; ■ иллюзия готовности ИС; ■ низкое качество проектной и эксплуатационной документации; ■ сосредоточенность на интерфейсе пользователя в ущерб проработке системных функций; ■ дублирование модулей; ■ несогласованность данных.

Границы применимости прототипирования

- Объем проекта и требования бизнеса четко определены, не изменяются, а сам проект невелик;
- проект не зависит от других средств автоматизации бизнеса, количество внешних интерфейсов ограничено;
- система ориентирована на экранные формы, обработка данных и системные функции составляют незначительную часть, удобство экранных форм является важнейшим фактором успеха проекта;
- пользователи имеют высокую квалификацию и изначально положительно оценивают идею создания новой системы.

Приобретение готового решения ИС

Достоинства	Недостатки
■ минимальные задержки и затраты до внедрения ИС; ■ возможность выбора пакета модулей, наиболее соответствующих требованиям организации; ■ возможность наглядной оценки функциональных возможностей готового продукта; ■ наличие полного пакета документации на ИС.	■ наличие вероятности того, что разработчик прекратит свое существование или обслуживание ИС; ■ отсутствие полного соответствия между возможностями готовых IT-продуктов и потребностями организации; ■ выбор и оценка готовых решений требуют дополнительных ресурсов.

Приобретение ядра ИС с последующей модификацией

Достоинства	Недостатки
■ уменьшение затрат ресурсов организации по сравнению с самостоятельной разработкой; ■ преодоление функциональных ограничений; ■ повышение степени удовлетворения потребностей организации.	■ возможность возникновения трудностей при модификации, что порождает новые ошибки и проблемы контроля внедрения; ■ усложнение процесса ведения документации по внесенным изменениям; ■ возможность отказа со стороны разработчика в обслуживании модифицированных решений.

Аренда ИС у ASP провайдера

- **Application Service Providing** – это технология, позволяющая создавать решения по предоставлению в аренду пользователю необходимого набора телекоммуникационных служб и приложений, на основе удаленного доступа к информационному комплексу, на котором установлено специальное программное обеспечение.

Задачи, решаемые с помощью ASP

- хостинг web - сайтов, почтовых служб;
- предоставление в аренду виртуальных торговых площадок для осуществления продаж/покупок через Интернет;
- обеспечение гибко настраиваемого доступа пользователей к различным функциям приложений;
- предоставление защищенного доступа к корпоративным данным;
- поддержка процессов электронного обмена данными;
- предварительная настройка компонентов ERP - систем на типовые задачи, что позволяет максимально сократить время внедрения таких систем в эксплуатацию;
- эксплуатация сложных ERP-систем

Типы ASP-решений

- Офисные и персональные приложения (Microsoft Office, игры, обучающие программы);
- Коммуникационные средства – электронная почта, проведение голосовых и видеоконференций, форум и т.д.;
- Приложения для электронной коммерции – электронные магазины, системы оплаты платежей;
- ERP-системы и отдельные приложения, например, CRM;
- Аналитические приложения – исследования и прогнозирование спроса, рисков и т.д.;
- Группы отраслевых приложений, представляющие собой специфические решения для определенных отраслей промышленности.

Аренда ИС у ASP провайдера

Достоинства	Недостатки
■ Более низкая стоимость за счет распределения стоимости ASP-решения на нескольких арендаторов; ■ гарантия фиксированной оплаты услуг; ■ круглосуточная техническая поддержка; ■ быстрое обновление оборудования.	■ обеспечение информационной безопасности; ■ обеспечение качественной бесперебойной связи; ■ ответственность провайдера услуг при остановке или сбоях в работе сервера за бизнес своих клиентов