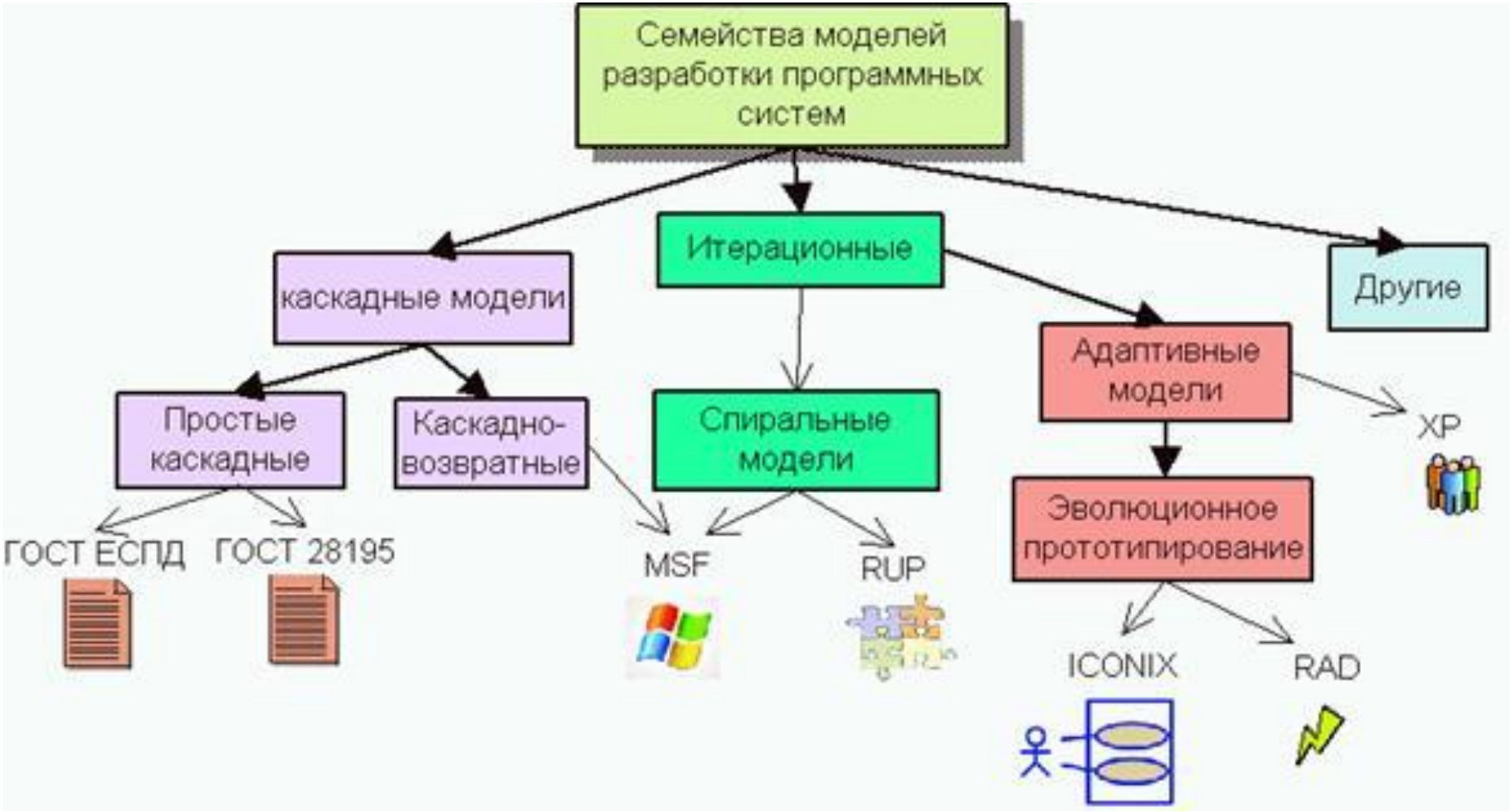
# МОДЕЛИ РАЗРАБОТКИ ПО

Тема 2.1

# Модели разработки ПО



# Семейство каскадных моделей:
## простая каскадная модель

# Принципы каскадной модели

- Строго последовательное выполнение фаз
  - Каждая последующая фаза начинается лишь тогда, когда полностью завершено выполнение предыдущей фазы
  - Каждая фаза имеет определенные критерии входа и выхода: входные и выходные данные.
  - Каждая фаза полностью документируется
  - Переход от одной фазы к другой осуществляется посредством
  - формального обзора с участием заказчика
- Основа модели – сформулированные требования (ТЗ), которые меняться не должны
- Критерий качества результата – соответствие продукта установленным требованиям

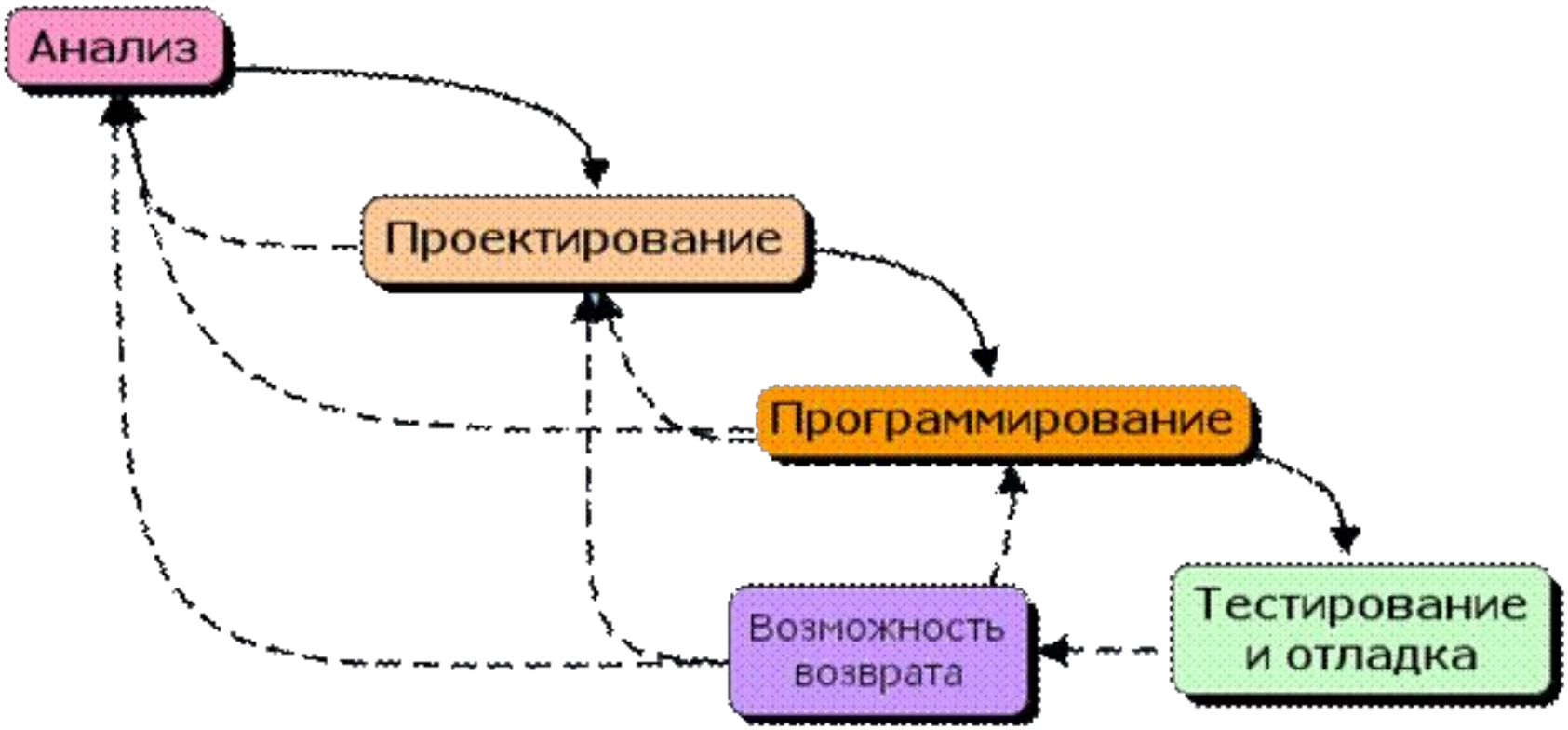
# Преимущества каскадной модели

- Проста и понятна заказчикам, т.к часто используется другими организациями для отслеживания проектов, не связанных с разработкой ПО
- Проста и удобна в применении:
  - процесс разработки выполняется поэтапно
  - ее структурой может руководствоваться даже слабо подготовленный в техническом плане или - неопытный персонал
  - она способствует осуществлению строгого контроля менеджмента проекта
- Каждую стадию могут выполнять независимые команды (все документировано)
- Позволяет достаточно точно планировать сроки и затраты

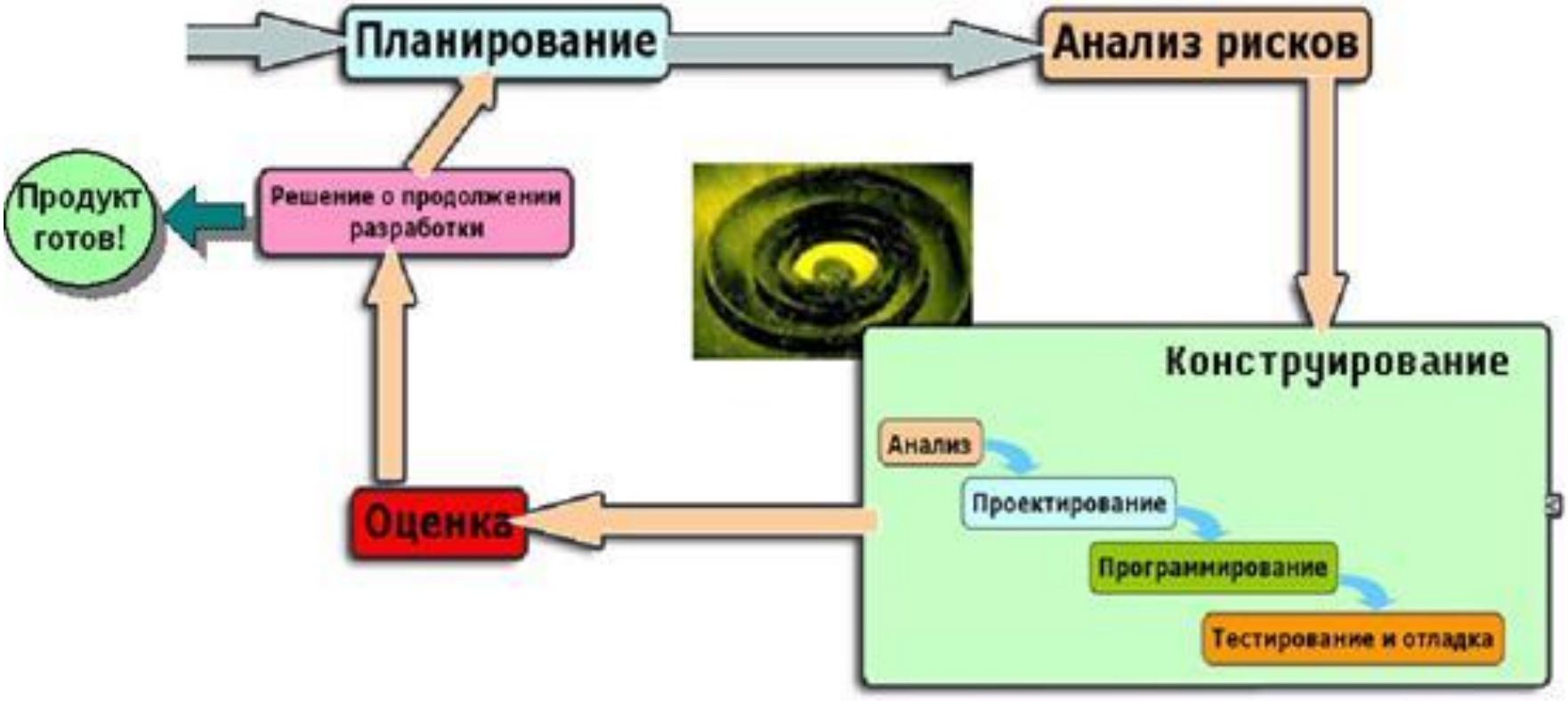
# Недостатки каскадной модели

- Попытка вернуться на одну или две фазы назад, чтобы исправить какую-либо проблему или недостаток, приведет к значительному увеличению затрат и сбою в графике
- Интеграция компонент, на которой обычно выявляется большая часть ошибок, выполняется в конце разработки, что сильно увеличивает стоимость устранения ошибок
- Запаздывание с получением результатов – если в процессе выполнения проекта требования изменились, то получится устаревший результат

# Семейство каскадных моделей:
## каскадно - возвратная модель



# Семейство итерационных моделей: спиральные модели



# Основные принципы спиральной модели

- Разработка вариантов продукта, соответствующих различным вариантам требований с возможностью вернуться к более ранним вариантам
- Создание прототипов ПО как средства общения с заказчиком для уточнения и выявления требований
- Планирование следующих вариантов с оценкой альтернатив и анализом рисков, связанных с переходом к следующему варианту
- Переход к разработке следующего варианта до завершения предыдущего в случае, когда риск завершения очередного варианта (прототипа) становится неоправданно высок.
- Использование каскадной модели как схемы разработки очередного варианта
- Активное привлечение заказчика к работе над проектом. Заказчик участвует в оценке очередного прототипа ПО, уточнении требований при переходе к следующему, оценке предложенных альтернатив очередного варианта и оценке рисков

# Спиральная разработка

# Преимущества спиральной модели

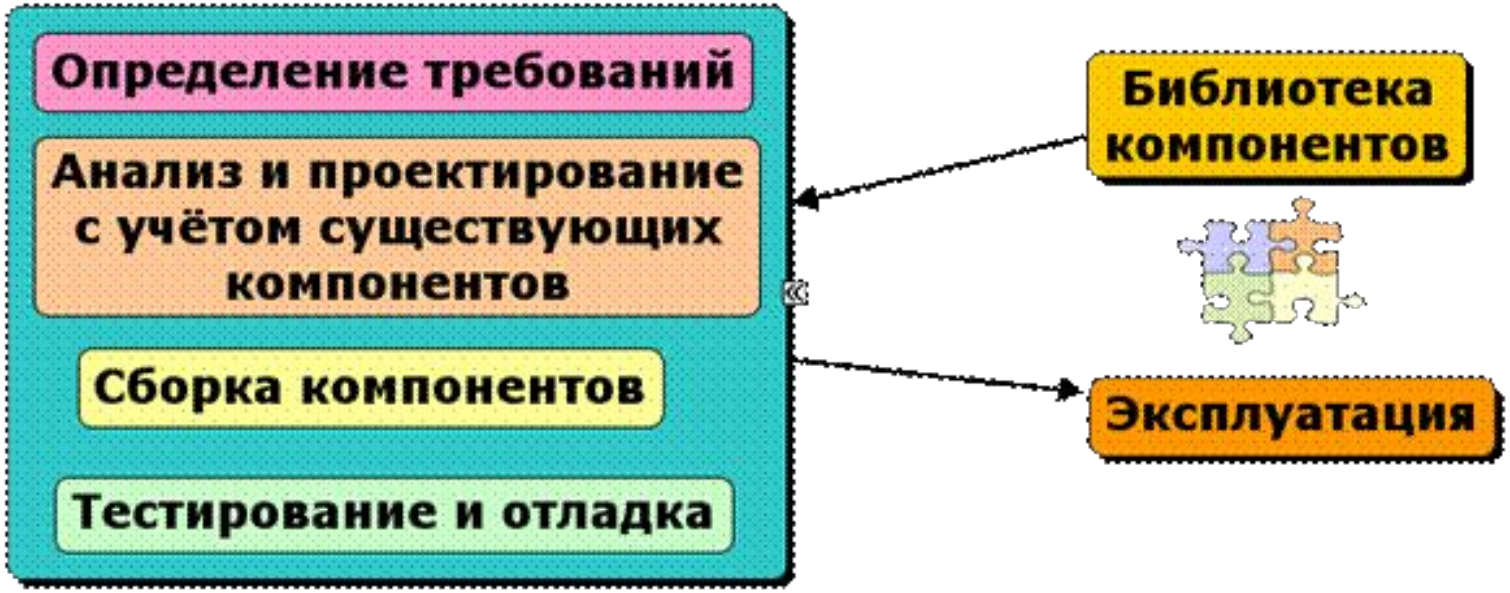
- Более тщательное проектирование (несколько начальных итераций) с оценкой результатов проектирования, что позволяет выявить ошибки проектирования на более ранних стадиях.
- Поэтапное уточнение требований в процессе выполнения итераций, что позволяет более точно удовлетворить требованиям заказчика
- Участие заказчика в выполнении проекта с использованием прототипов программы. Заказчик видит, что и как создается, не выдвигает необоснованных требований, оценивает реальные объемы финансирования
- Планирование и управление рисками при переходе на следующие итерации позволяет разумно планировать использование ресурсов и обосновывать финансирование работ.
- Возможность разработки сложного проекта «по частям», выделяя на первых этапах наиболее значимые требования.

# Недостатки спиральной модели

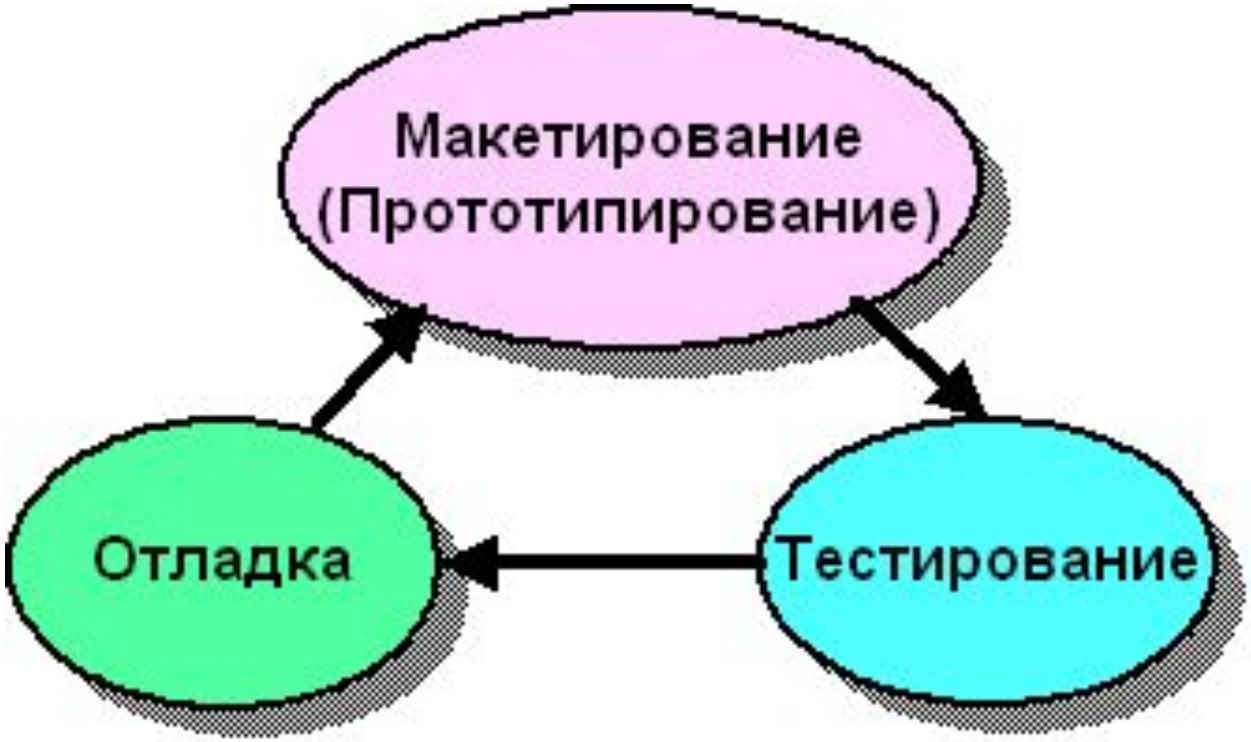
- Сложность анализа и оценки рисков при выборе вариантов.
- Сложность поддержания версий продукта (хранение версий, возврат к ранним версиям, комбинация версий)
- Сложность оценки точки перехода на следующий цикл
- Бесконечность модели – на каждом витке заказчик может выдвигать новые требования, которые приводят к необходимости следующего цикла разработки

# Семейство итерационных моделей: каркасная модель разработки

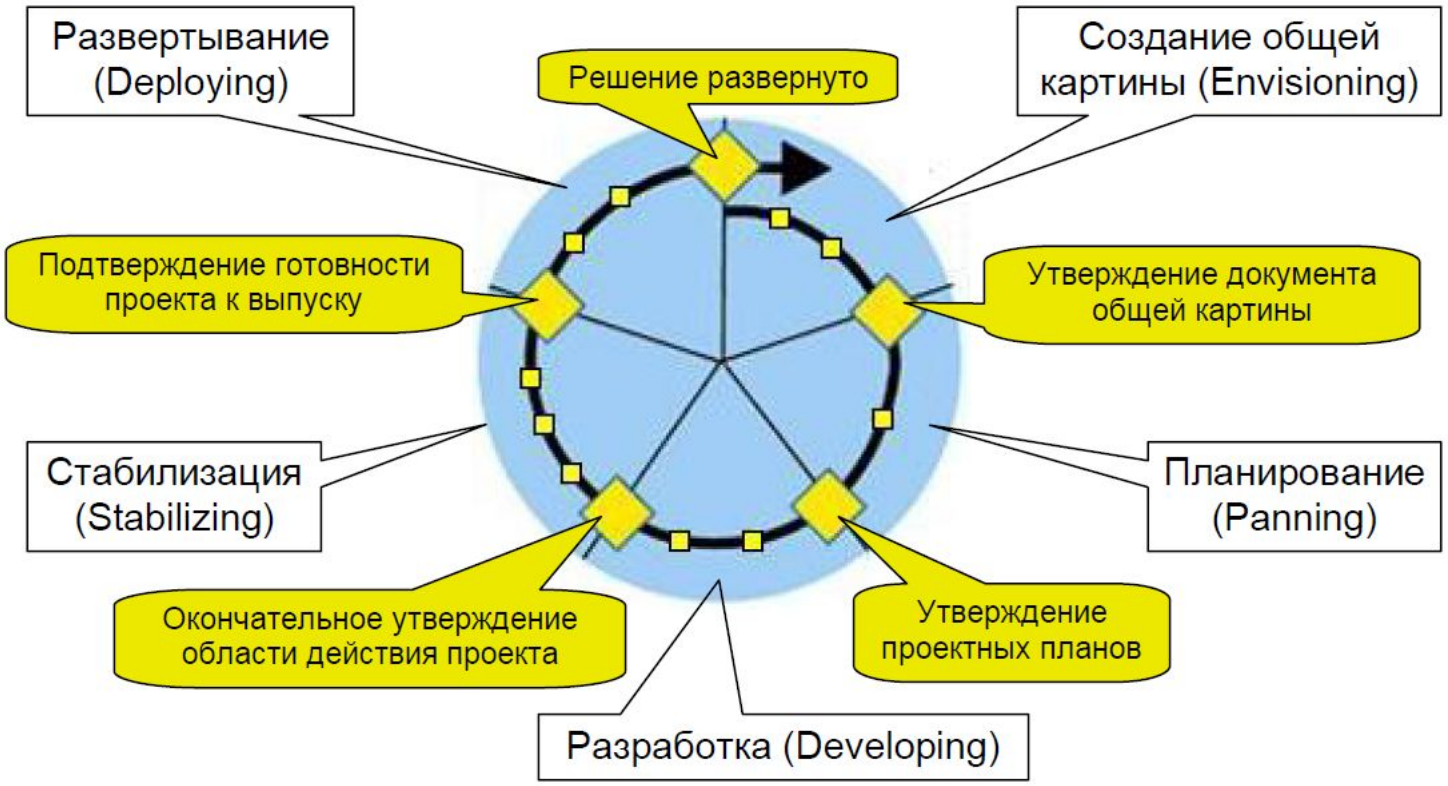
# Другие модели: сборочное программирование



# Другие модели: исследовательское программирование



# Модель разработки Microsoft Solution Framework




В технологии MSF большое внимание уделяется анализу проблем заказчика и разработке вариантов системы для поиска решения этих проблем

# «Вехи» MSF

- Модель MSF ориентирована на “вехи” (milestones) – ключевые точки проекта, характеризующие достижение какого-либо существенного результата
- 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

# Фазы MSF: выработка концепции

- Создание общей картины приложения (Envisioning).На этом этапе решаются следующие основные 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".

# Фазы MSF: Планирование (Planning)

- Включает планирование и проектирование продукта.
- На основе анализа требований разрабатывается проект и основные архитектурные решения, функциональные спецификации системы, планы и календарные графики, 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.
- На стадии **физического проектирования** задача рассматривается с точки зрения программистов, уточняются используемы е технологии и интерфейсы.

# Фазы MSF: 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.
- 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.

# Фазы MSF: Стабилизация (Stabilizing)

- Подготовка к выпуску окончательной версии продукта, доводка его до заданного уровня качества.
- Выполняется комплекс работ по тестированию (обнаружение и устранение дефектов), проверяется сценарий развертывания продукта. Когда решение становится достаточно устойчивым, проводится его пилотная эксплуатация в тестовой среде с привлечением пользователей и применением реальных сценариев работы.

# Фазы MSF: Развертывание (Deploying)

- Выполняется установка решения и необходимых компонентов окружения, проводится его стабилизация в промышленных условиях и передача проекта в руки группы сопровождения.
- Анализируется проект в целом на предмет уровня удовлетворенности заказчика.

# Ключевые идеи RUP

- весь ход работ направляется итоговыми целями проекта, выраженными в виде вариантов использования (use cases) – сценариев взаимодействия результирующей программной системы с пользователями или другими системами, при выполнении которых пользователи получают значимые для них результаты и услуги. Разработка начинается с выделения вариантов использования и на каждом шаге контролируется степенью приближения к их реализации;
- основным решением, принимаемым в ходе проекта, является архитектура результирующей программной системы. Архитектура устанавливает набор компонентов, из которых будет построено ПО, ответственность каждого из компонентов (т.е. решаемые им подзадачи в рамках общих задач системы), четко определяет интерфейсы, через которые они могут взаимодействовать, а также способы взаимодействия компонентов друг с другом; архитектура является одновременно основой для получения качественного ПО и базой для планирования работ и оценок проекта в терминах времени и ресурсов, необходимых для достижения определенных результатов. Она оформляется в виде набора графических моделей на языке UML;
- основой процесса разработки являются планируемые и управляемые итерации, объем которых (реализуемая в рамках итерации функциональность и набор компонентов) определяется на основе архитектуры.

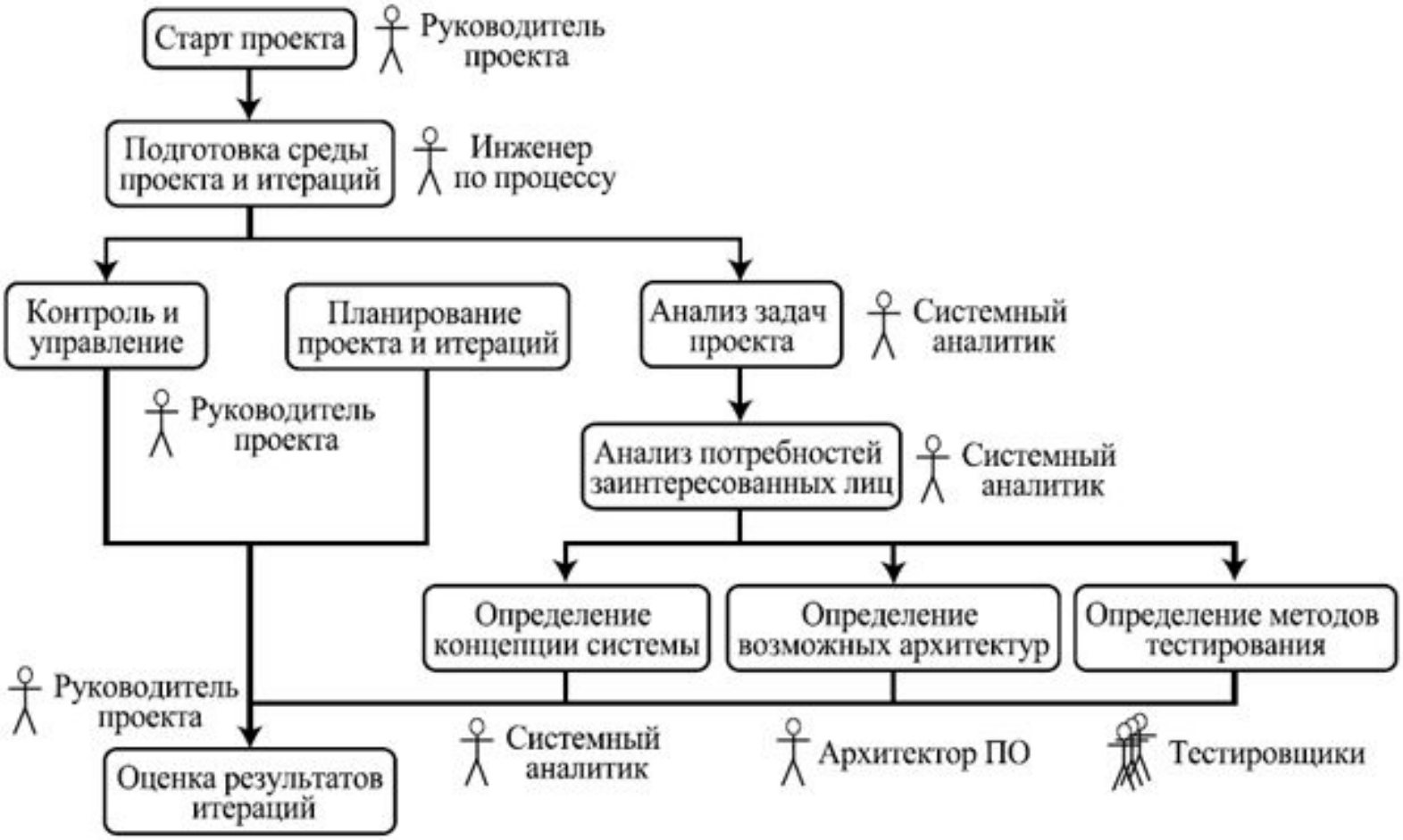
# Фазы жизненного цикла RUP

- *Фаза начала проекта* (Inception)
- *Фаза проектирования* (Elaboration)
- *Фаза построения* (Construction)
- *Фаза внедрения* (Transition)

# Фаза начала проекта

- Основная цель этой фазы – достичь компромисса между всеми заинтересованными лицами относительно задач проекта и выделяемых на него ресурсов.
- На этой стадии определяются основные цели проекта, руководитель и бюджет, основные средства выполнения – технологии, инструменты, ключевые исполнители. Также, возможно, происходит апробация выбранных технологий, чтобы убедиться в возможности достичь целей с их помощью, и составляются предварительные планы проекта.
- На эту фазу может уходить около 10% времени и 5% трудоемкости одного цикла.

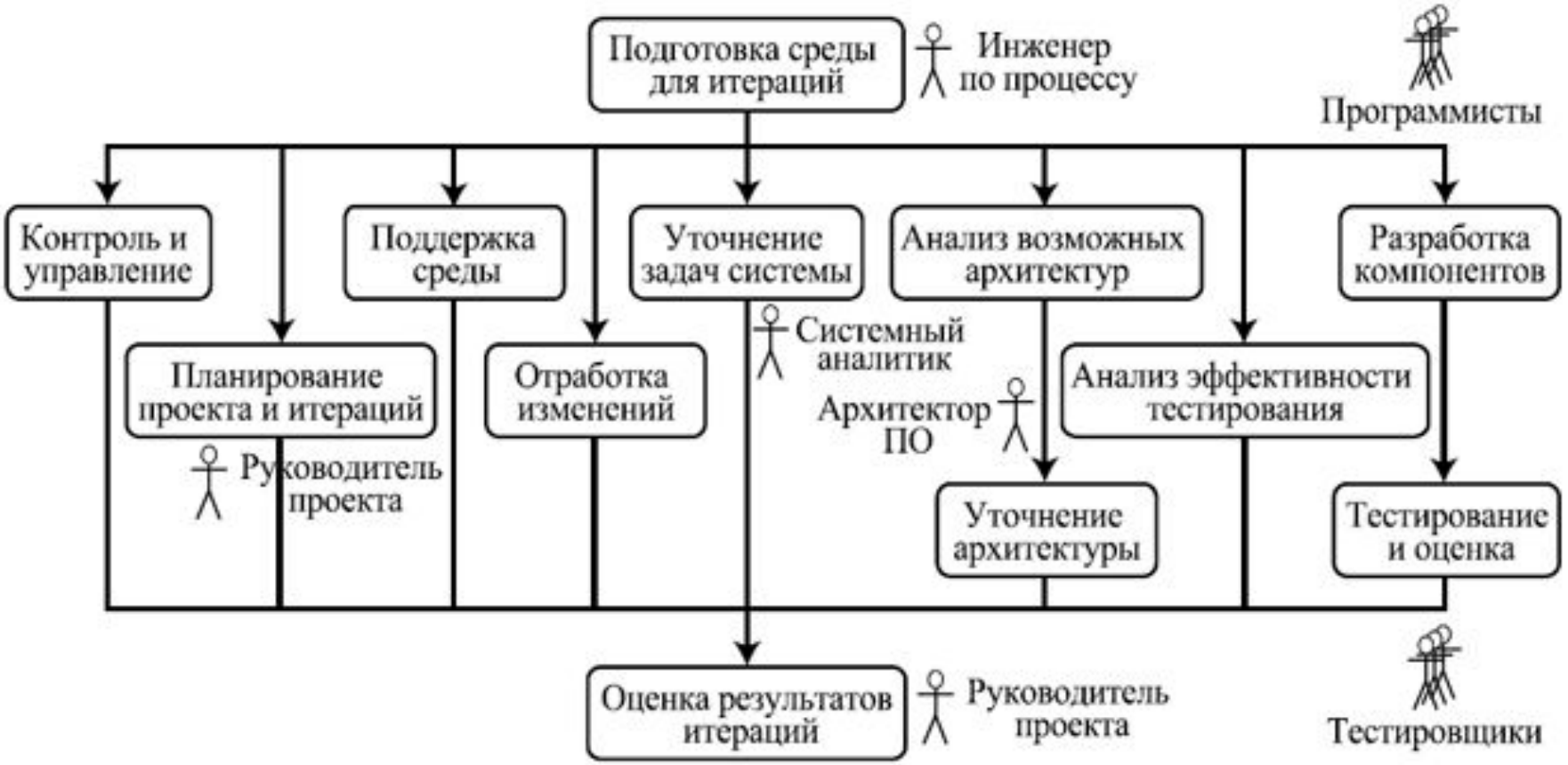
# Пример хода работ на фазе начала проекта



# Фаза проектирования

- Основная цель этой фазы – на базе основных, наиболее существенных требований разработать стабильную базовую архитектуру продукта, которая позволяет решать поставленные перед системой задачи и в дальнейшем используется как основа разработки системы.
- На эту фазу может уходить около 30% времени и 20% трудоемкости одного цикла.

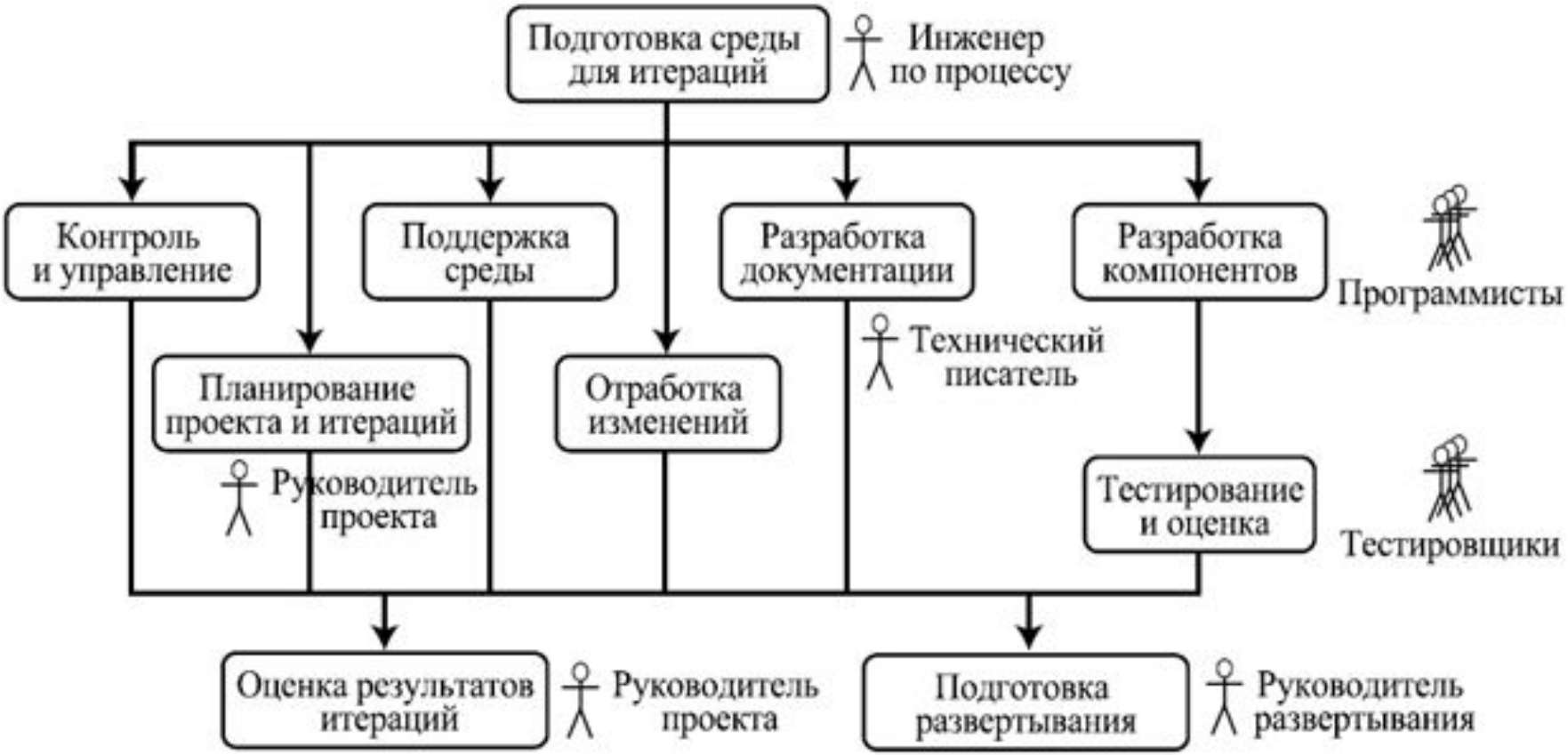
# Пример хода работ на фазе проектирования



# Фаза построения

- Основная цель этой фазы – детальное прояснение требований и разработка системы, удовлетворяющей им, на основе спроектированной ранее архитектуры. В результате должна получиться система, реализующая все выделенные варианты использования.
- На эту фазу уходит около 50% времени и 65% трудоемкости одного цикла.

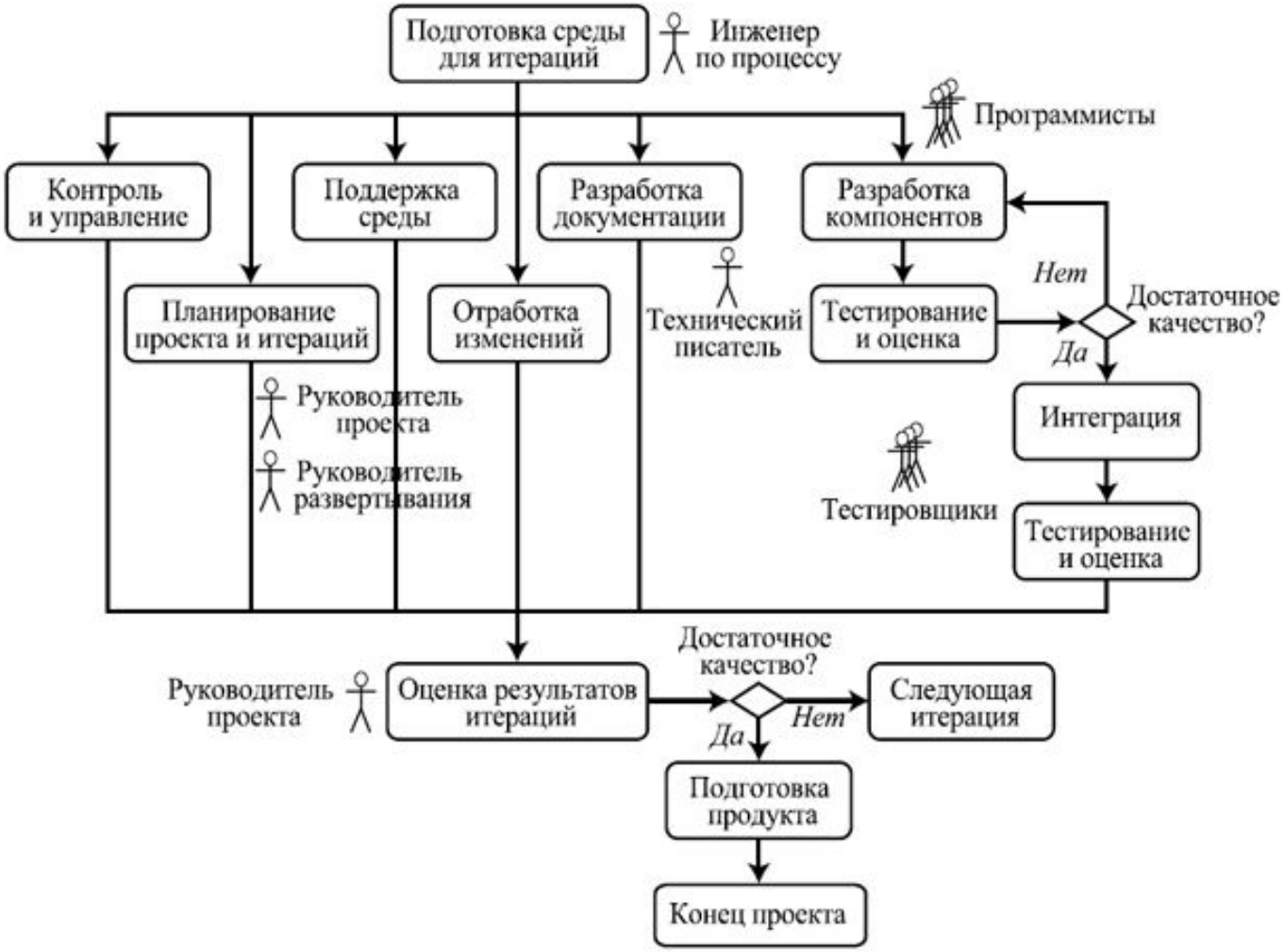
# Пример хода работ на фазе построения



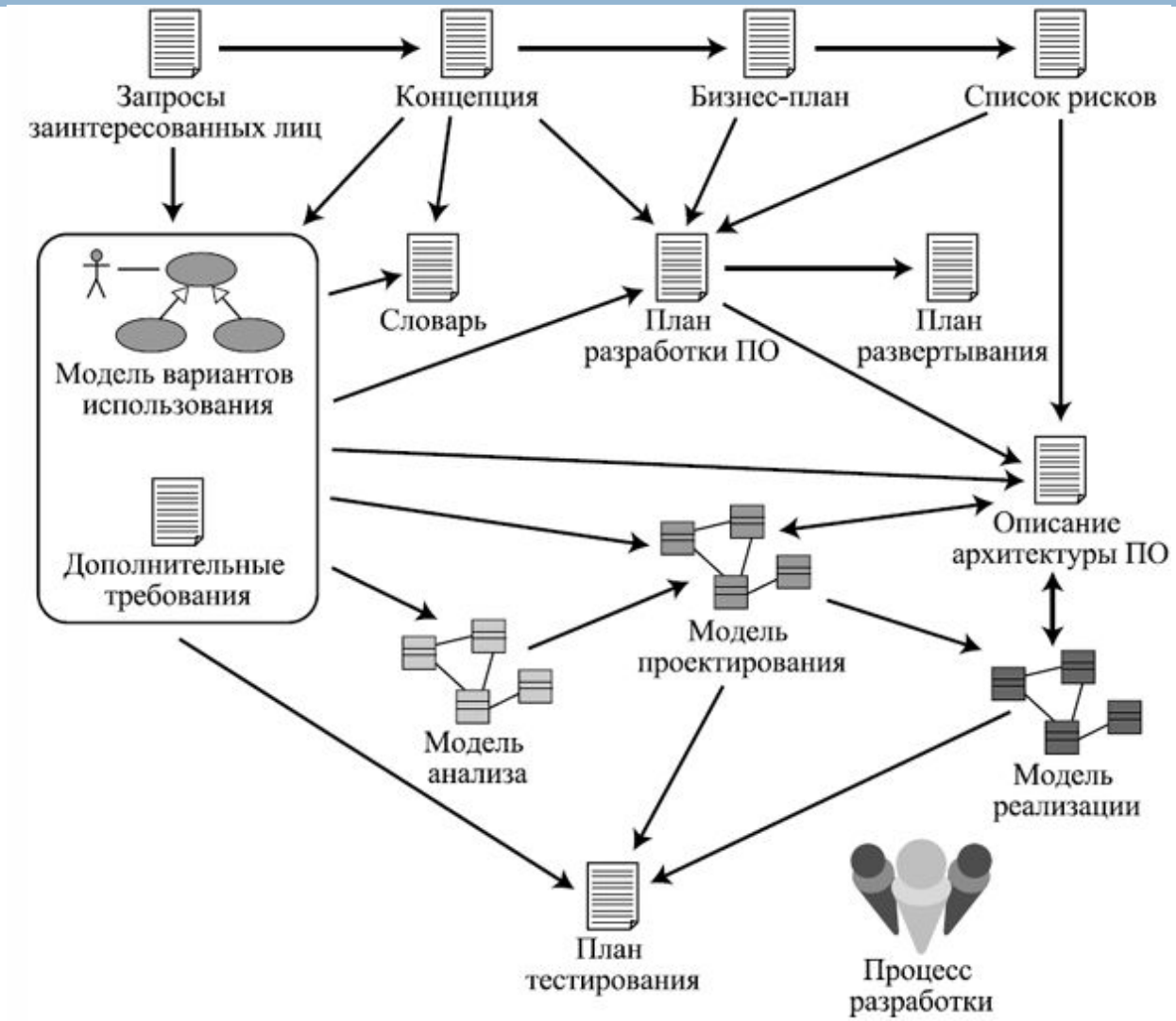
# Фаза внедрения

- Цель этой фазы – сделать систему полностью доступной конечным пользователям. На этой стадии происходит развертывание системы в ее рабочей среде, бета-тестирование, подгонка мелких деталей под нужды пользователей.
- На эту фазу может уходить около 10% времени и 10% трудоемкости одного цикла.

# Пример хода работ на фазе внедрения



# Основные артефакты проекта по RUP и потоки данных между ними



# Дисциплины RUP: определение

- **Дисциплины** включают различные наборы деятельностей, которые в разных комбинациях и с разной интенсивностью выполняются на разных фазах.
- В документации по процессу каждая дисциплина сопровождается довольно большой диаграммой, поясняющей действия, которые нужно выполнить в ходе работ в рамках данной дисциплины, артефакты, с которыми надо иметь дело, и роли вовлеченных в эти действия лиц.

# Рабочие дисциплины RUP

*Моделирование предметной области* (бизнес-моделирование, Business Modeling)

Задачи этой деятельности – понять предметную область или бизнес-контекст, в которых должна будет работать система, и убедиться, что все заинтересованные лица понимают его одинаково, осознать имеющиеся проблемы, оценить их возможные решения и их последствия для бизнеса организации, в которой будет работать система.

В результате моделирования предметной области должна появиться ее модель в виде набора диаграмм классов (объектов предметной области) и деятельностей (представляющих бизнес-операции и бизнес-процессы). Эта модель служит основой модели анализа.

*Определение требований* (Requirements)

Задачи – понять, что должна делать система, и убедиться во взаимопонимании по этому поводу между заинтересованными лицами, определить границы системы и основу для планирования проекта и оценок затрат ресурсов в нем.

Требования принято фиксировать в виде модели вариантов использования.

*Анализ и проектирование* (Analysis and Design)

Задачи – выработать архитектуру системы на основе требований, убедиться, что данная архитектура может быть основой работающей системы в контексте ее будущего использования.

В результате проектирования должна появиться модель проектирования, включающая диаграммы классов системы, диаграммы ее компонентов, диаграммы взаимодействий между объектами в ходе реализации вариантов использования, диаграммы состояний для отдельных объектов и диаграммы развертывания.

*Реализация* (Implementation)

Задачи – определить структуру исходного кода системы, разработать код ее компонентов и протестировать их, интегрировать систему в работающее целое.

*Тестирование* (Test)

Задачи – найти и описать дефекты системы (проявления недостатков ее качества), оценить ее качество в целом, оценить, выполнены или нет гипотезы, лежащие в основе проектирования, оценить степень соответствия системы требованиям.

# Поддерживающие дисциплины

*Развертывание* (Deployment)

Задачи – установить систему в ее рабочем окружении и оценить ее работоспособность на том месте, где она должна будет работать.

*Управление конфигурациями и изменениями* (Configuration and Change Management)

Задачи – определение элементов, подлежащих хранению в репозитории проекта и правил построения из них согласованных конфигураций, поддержание целостности текущего состояния системы, проверка согласованности вносимых изменений.

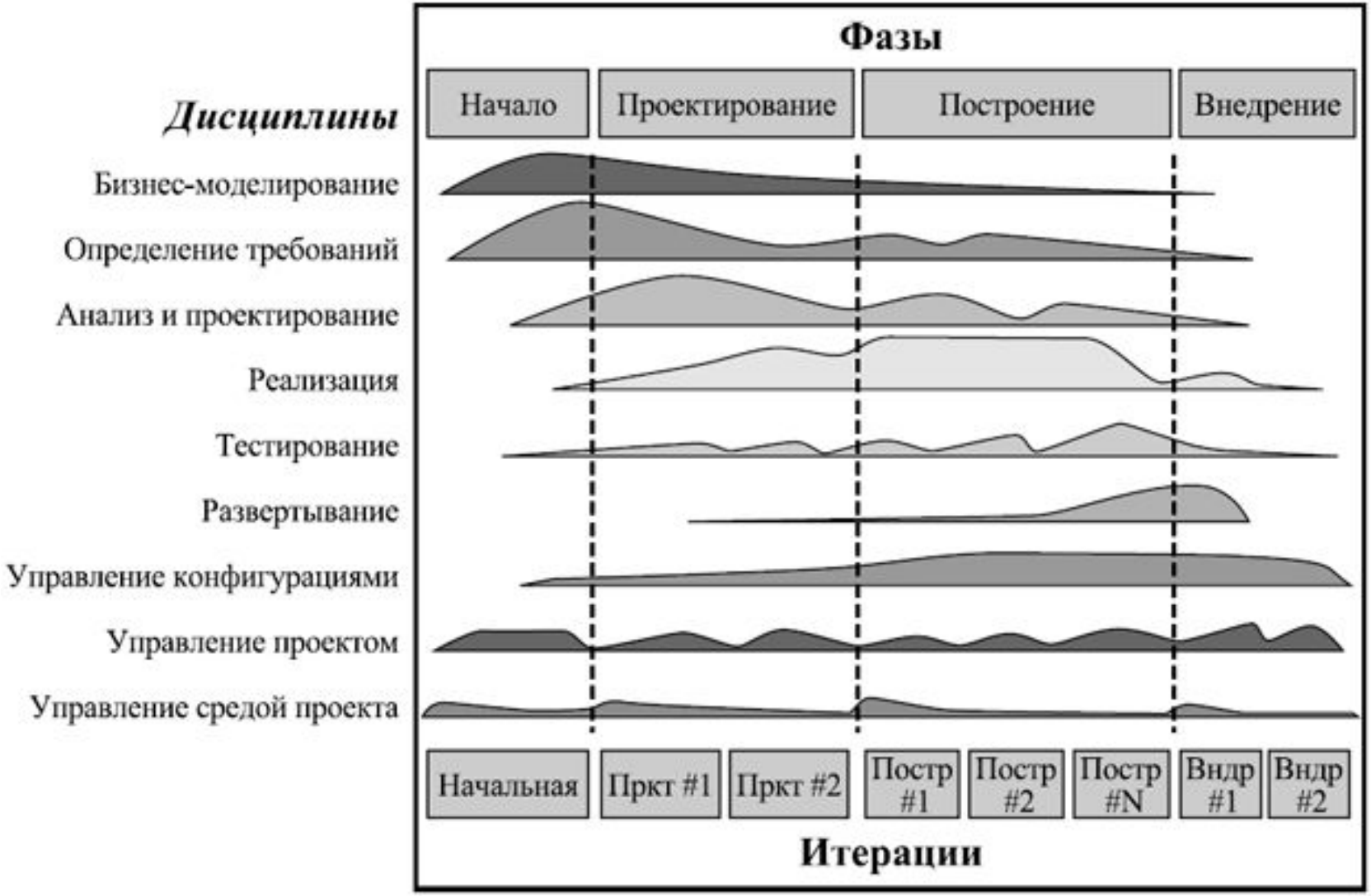
*Управление проектом* (Project Management)

Задачи – планирование, управление персоналом, обеспечение взаимодействия на благо проекта между всеми заинтересованными лицами, управление рисками, отслеживание текущего состояния проекта.

*Управление средой проекта* (Environment)

Задачи – подстройка процесса под конкретный проект, выбор и замена технологий и инструментов, используемых в проекте.

# Распределение работ между различными дисциплинами в проекте по RUP



# Экстремальное программирование

- возникло как эволюционный метод разработки ПО "снизу вверх"
- является примером так называемого метода "живой" разработки (Agile Development Method)
- в группу "живых" методов входят, помимо экстремального программирования, методы SCRUM, DSDM (Dynamic Systems Development Method, метод разработки динамичных систем), Feature-Driven Development (разработка, управляемая функциями системы) и др.

# Основные принципы "живой" разработки ПО

- Люди, участвующие в проекте, 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.

# Схема потока работ в XP

# Достоинства и недостатки

- Достоинствами XP, если его удается применить, является большая гибкость, возможность быстро и аккуратно вносить изменения в ПО в ответ на изменения требований и отдельные пожелания заказчиков, высокое качество получающегося в результате кода и отсутствие необходимости убеждать заказчиков в том, что результат соответствует их ожиданиям.
- Недостатками этого подхода являются невыполнимость в таком стиле достаточно больших и сложных проектов, невозможность планировать сроки и трудоемкость проекта на достаточно долгую перспективу и четко предсказать результаты длительного проекта в терминах соотношения качества результата и затрат времени и ресурсов. Также можно отметить неприспособленность XP для тех случаев, в которых возможные решения не находятся сразу на основе ранее полученного опыта, а требуют проведения предварительных исследований.