HEICA

TEMPUS

Процессы Разработки программного обеспечения

Определение процесса

(Часть 1)

HEICA

TEMPUS

Цели

Цель:

- Ознакомится с некоторыми моделями процессов разработки программного обеспечения
- Узнать функции использования этих моделей
- Сравнение этих моделей между собой
- Определить, когда уместно применить эти модели

Акронимы

- SW – программное обеспечение
- SS – Система (ы) программного обеспечения
- UC – Вариант (ы) использования

2. Определение процесса

Процесс = технология + инструменты + люди + модель (1)

- Технология – в разработке SS используется доступные среды, знание, правила...
  (языки программирования, ОС, компьютерные системы, сети, CASE инструменты...)
- Инструменты –предварительно разработанные инструменты для разработки SS, которые сохраняются и улучшаются во время процесса вместе с основной разработкой продукта
  (чем лучше ведущее устройство (мастер), тем более специальные инструменты он имеет, например. NoMagic)

Процесс = технология + инструменты + люди + модель (2)

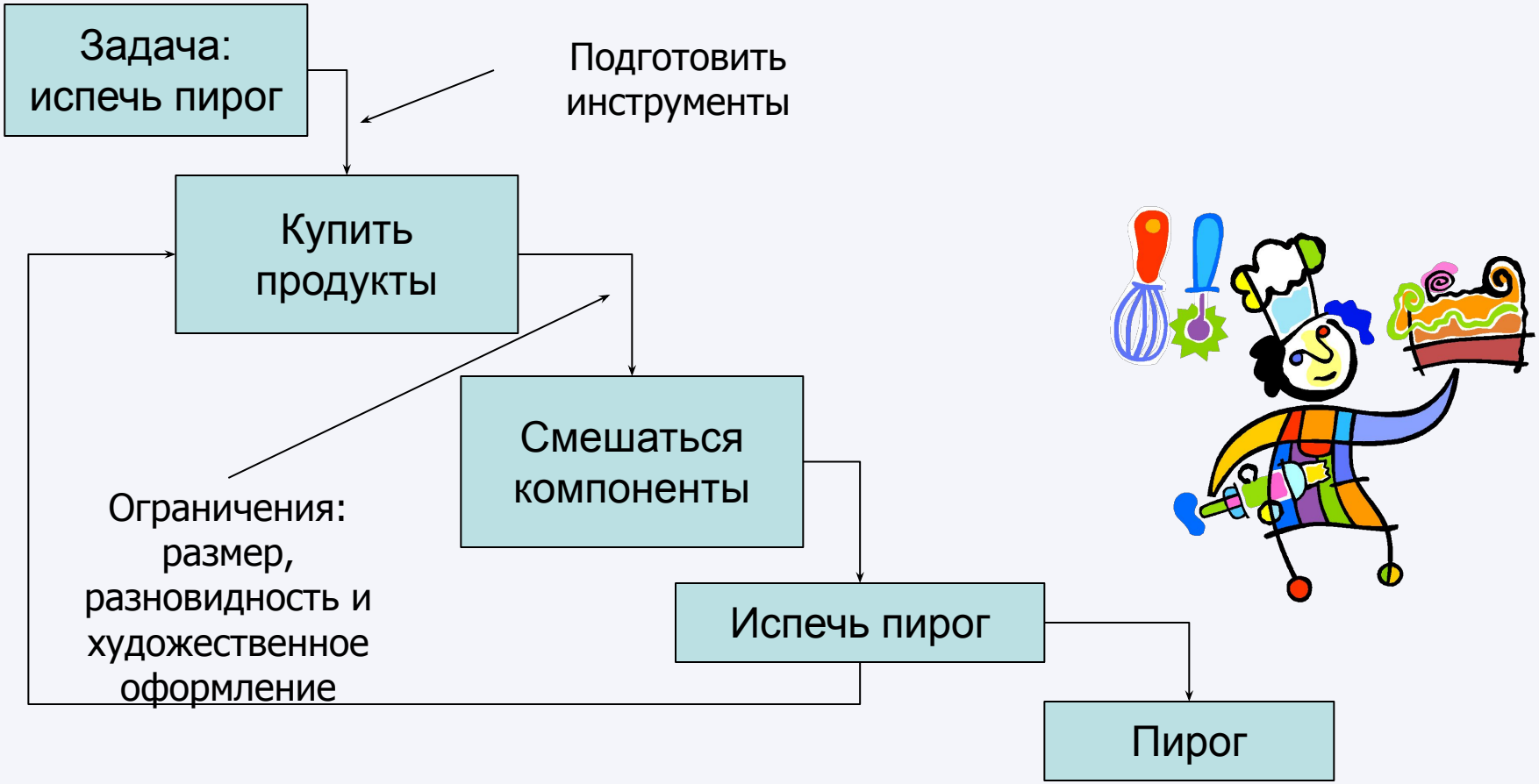
- Люди – участники процесса и пользователи SS

  (необходимо определить требования разработчиков SS, квалификацию команды, управлять ими и пользовательскими навыками; зависимости требований SS, квалификации разработчиков и полной компьютерной грамотности людей; наиболее ограниченный параметр)

- Организационная модель – определение, кто, что, когда и как делает + организационные инструменты, стандарты и стратегии

  (Разработчик SS – не импровизирующий музыкант, он "играет примечания" – организационная модель проекта, который определяет, кто, что, когда и как должен сделать),

Каков процесс (кратко)?



Характеристики процесса (1)

Процесс

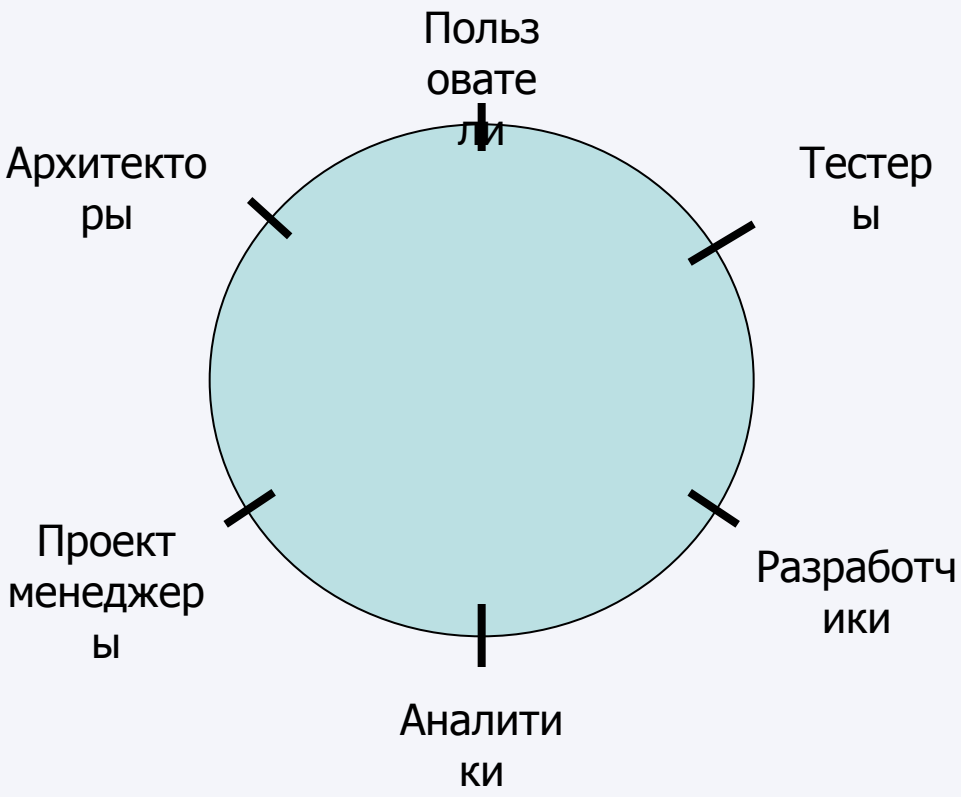
- Описывает основные виды деятельности согласно модели
- Имеет ряд принципов, которые объясняют цели действий
- Предлагает ограничения, выделяет промежуточные результаты и окончательный результат
- Может быть составлен из взаимозависимых подпроцессов (иерархия процессов, у каждого подпроцесса может быть своя собственная модель процесса),
- У действий есть критерии оценки для входа и выхода
- Знать последовательность когда начинаются и заканчиваются действия

Характеристики процесса (2)

- Когда процесс включает разработку SW, текущее его возобновление, обслуживание конкурентоспособности и живучести во время всей его жизни, это называется **процессом жизненных циклов**
- Процесс разработки SW во время одного цикла вызывают как **процесс жизненного цикла**, потому что это описывает жизнь SW от идеи до его установки, поставки, использования и обслуживания (первый процесс), или это описывает обновление SW, поддерживающего конкурентоспособность и живучесть (другие процессы)

Модели (1)

- Каждый сотрудник и этап из работ требуют индивидуальную модель
- Модель системы – это сумма отдельных моделей



Модели (2)

- Процесс разработки SS – процесс развития моделей. Объединенный процесс имеет ряд выбранных моделей, которые описывают SS всем разработчикам.
- Модель – абстракция системы, которая описывает ее в некотором аспекте. Каждая модель разъединяет себя с незначительными деталями этого этапа, и описывает важные аспекты этого этапа.
- Все модели описывают систему

HEICA

TEMPUS

Модели (3)

- Каждая модель является семантически завершенной абстракцией системы, достаточной, чтобы гарантировать, что ее клиентам не не надо искать дополнительную информацию для соответствующей интерпретации. Модели объединяют элементов среды – агентов.
- Каждая модель состоит из моделей подсистем. Уровень разработки неограничен
- В развитии моделей их части сохраняют отношение с частями для более ранних моделей
- Модели имеют разные уровни и разрабатываются в различных уровнях.

Модели (4) (Запомнить)

- Модель варианта использования системы состоит из вариантов и агентов
- Модель проекта описывает подсистемы, классы и их взаимодействия. В ней рассмотрены варианты использования. Каждый вариант использования модели проекта сохраняет отношения с вариантами использования предыдущей модели.
- Модель варианта использования – внешний подход (пользователя), модель проекта – внутренний подход (разработчика). Обе модели описывают, как система реагирует на внешние влияния, но в различных аспектах.

HEICA

TEMPUS

Определения

- Жизненный цикл – этап жизни определенного SW определенного проекта, от его идеи, разработки..., до изменения (удаление)
- Первый цикл – разработка первой версии (рождение), другие циклы (разработка, расширение), последний цикл – удаление (смерть)
- Модель жизненного цикла – система действий, предназначенных для жизненного цикла SW, выбранного из стандарта “1074 IEEEСтанд.”
- Процесс жизненного цикла – спецификация проекта процесса

2.1. Модели жизненного цикла разработки SS

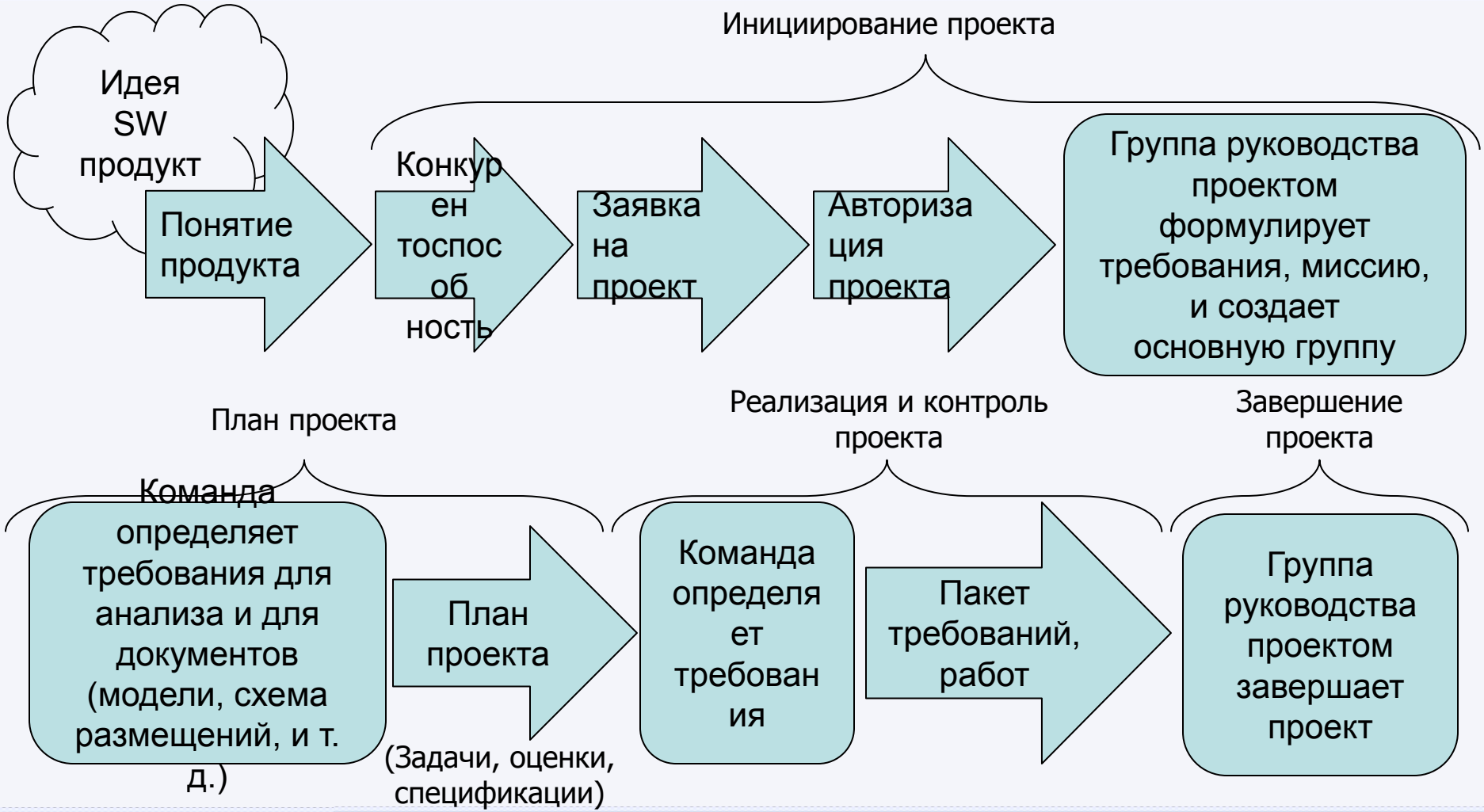
Модель технического жизненного цикла SS (1)

- Цель
- Представить процесс разработки SW и выделить области основного вида деятельности и их взаимозависимости
- Модели жизненного цикла могут быть применены:
  - Как инструмент для помощи людям, чтобы понять и вообразить процесс (особенно в графических моделях)
  - Как основание процесса разработки (разработка, оценка, улучшение)
  - Для формализации процесса, сосредотачиваясь на среде разработки SW или рабочем ходу

Модель технического жизненного цикла SS (2)

- Обычно в практике модели процесса используются:
  - Попытка описать процессы, оценить их с соответствующими характеристиками и возможностями улучшения
  - Разработка и анализ новых процессов
  - Количественное моделирование и анализ процессов для обслуживания, планирования управления и управления, и улучшения процесса

Общее определение модели технического жизненного цикла SS



Основные этапы процессов жизненного цикла SS

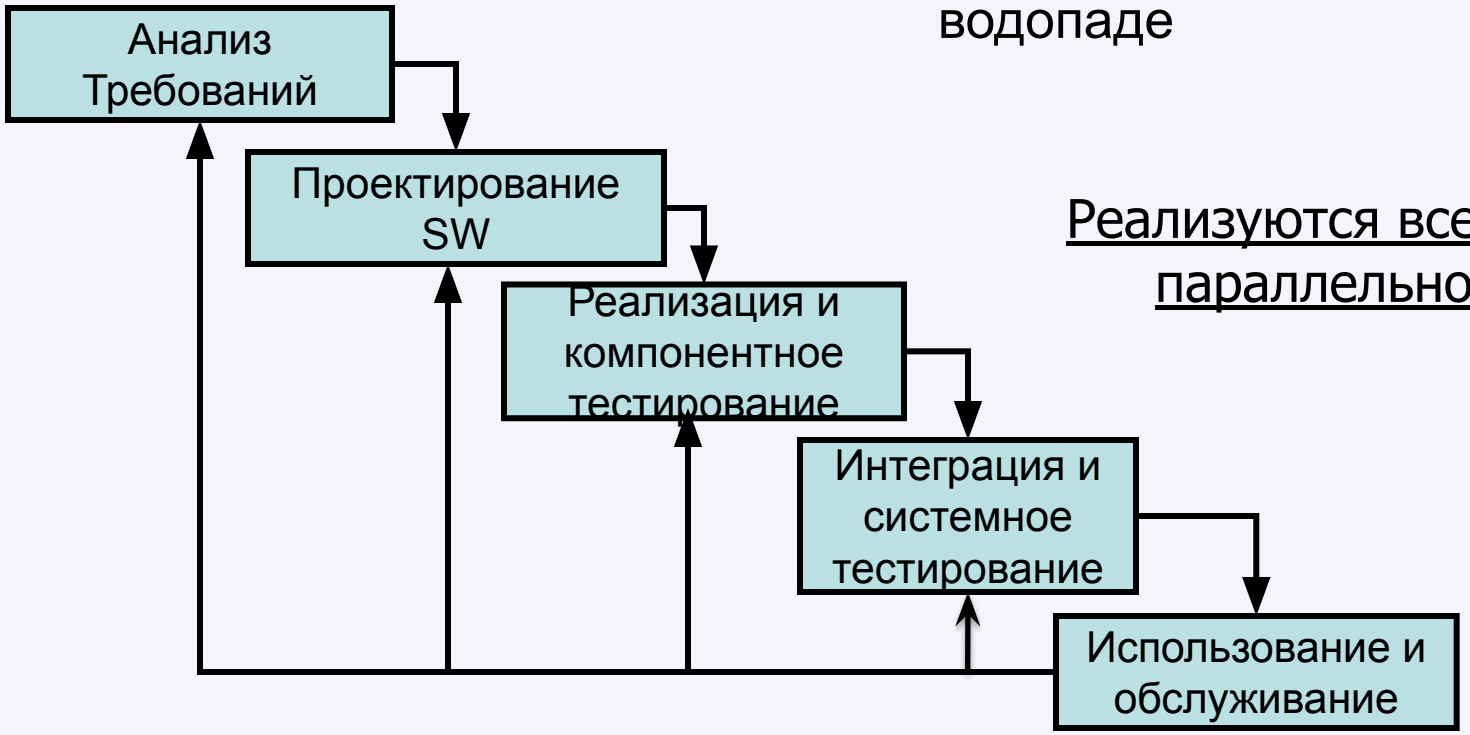
- Основные этапы:
  - Формулировка требований
  - Анализ
  - Разработка
  - Реализация
  - Тестирование
  - Эксплуатация
- Эти процессы циклически повторяются, разрабатывая, поддерживая и улучшая продукт

Модели разработки SS жизненный цикл

- Модели процессов:
  - Водопадная
  - Прототипирование
  - Эволюционная разработка
  - Формальный подход
  - Автоматизированный синтез программного обеспечения
  - Компонентная (допускающее повторное использование программного обеспечения)
- Модели организации в виде итераций:
  - Инкрементная
  - Итеративная

Водопадная (каскадная) модель

Процесс переходит от фазы в фазу последовательно как вода в каскадном водопаде



Реализуются все UC параллельно

Преимущества водопадной модели

1. Низкие трудовые затраты при небольшом количестве возвратов
2. Процесс является четким и легко скоординированным

- Уместно применить модель когда:
  - Требования пользователя являются четкими
  - Задача является традиционной
  - Алгоритм решения является четким
  - Задача не является трудной

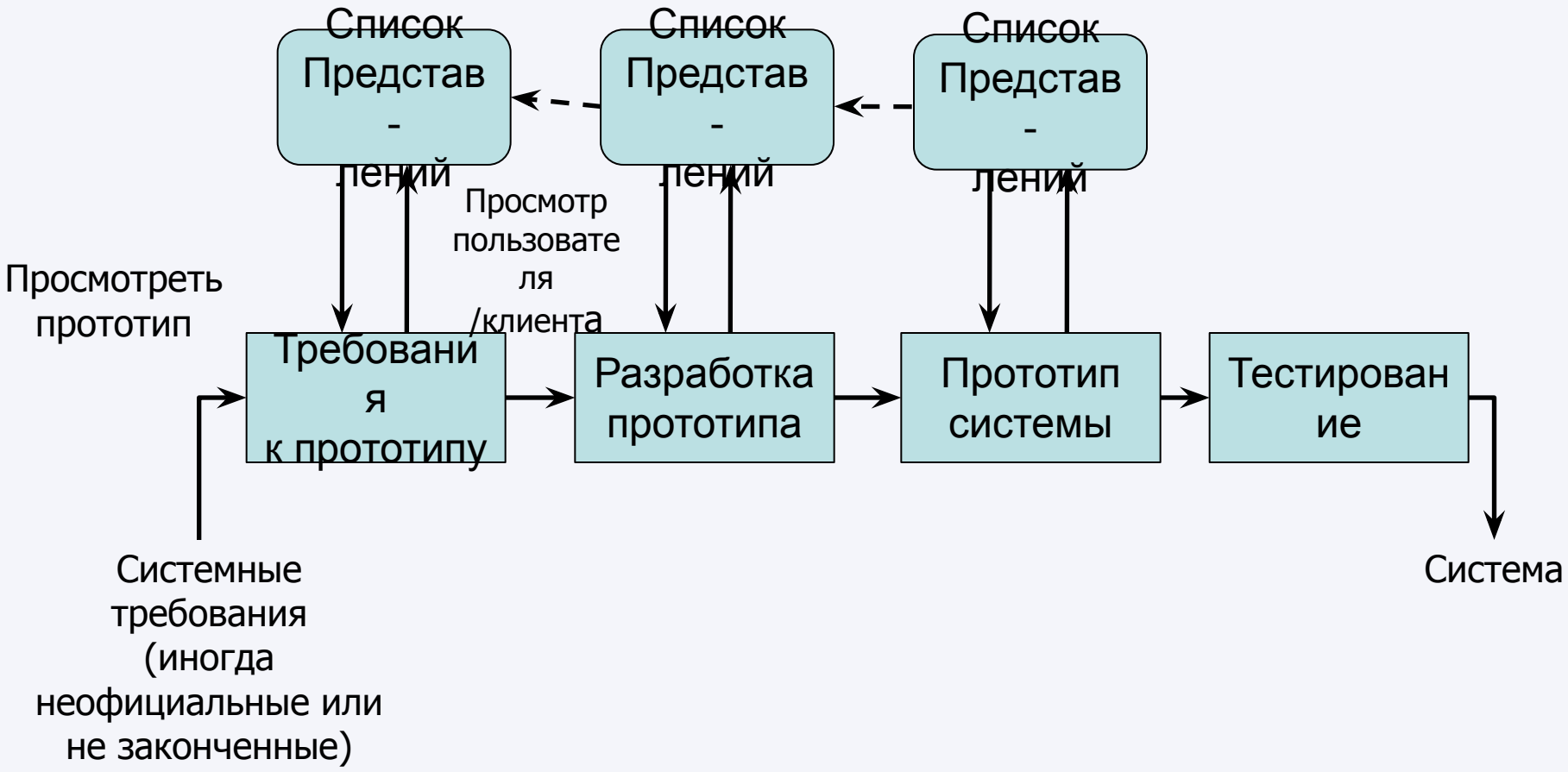
Недостатки водопадной модели

1. Негибкое распределение проекта по этапам.
2. Есть трудности, когда необходимо исправить ошибки или ответить на изменение требований пользователя.
3. Трудно произвести изменения во время процесса. Они обычно производятся после выполнения последней фазы.
4. Невозможно выдать пользователю какую-либо промежуточную версию SS
5. Начало эксплуатации возможно только тогда, когда все UC реализованы. Это соответствует самому длинному пути в временного графика.

Прототипирование (1)

- Это - системная модель разработки, когда система разрабатывается на основе эволюционных прототипов; прототип создается, тестируется и улучшается, пока не получается опытно-промышленный образец, который преобразовывается в заключительную систему
- Эта модель является самой подходящей, если вначале не все требования известны. Это основано на повторениях, попытках и ошибках в процессе, который имеет место между разработчиками и пользователями

Прототипирование (2)



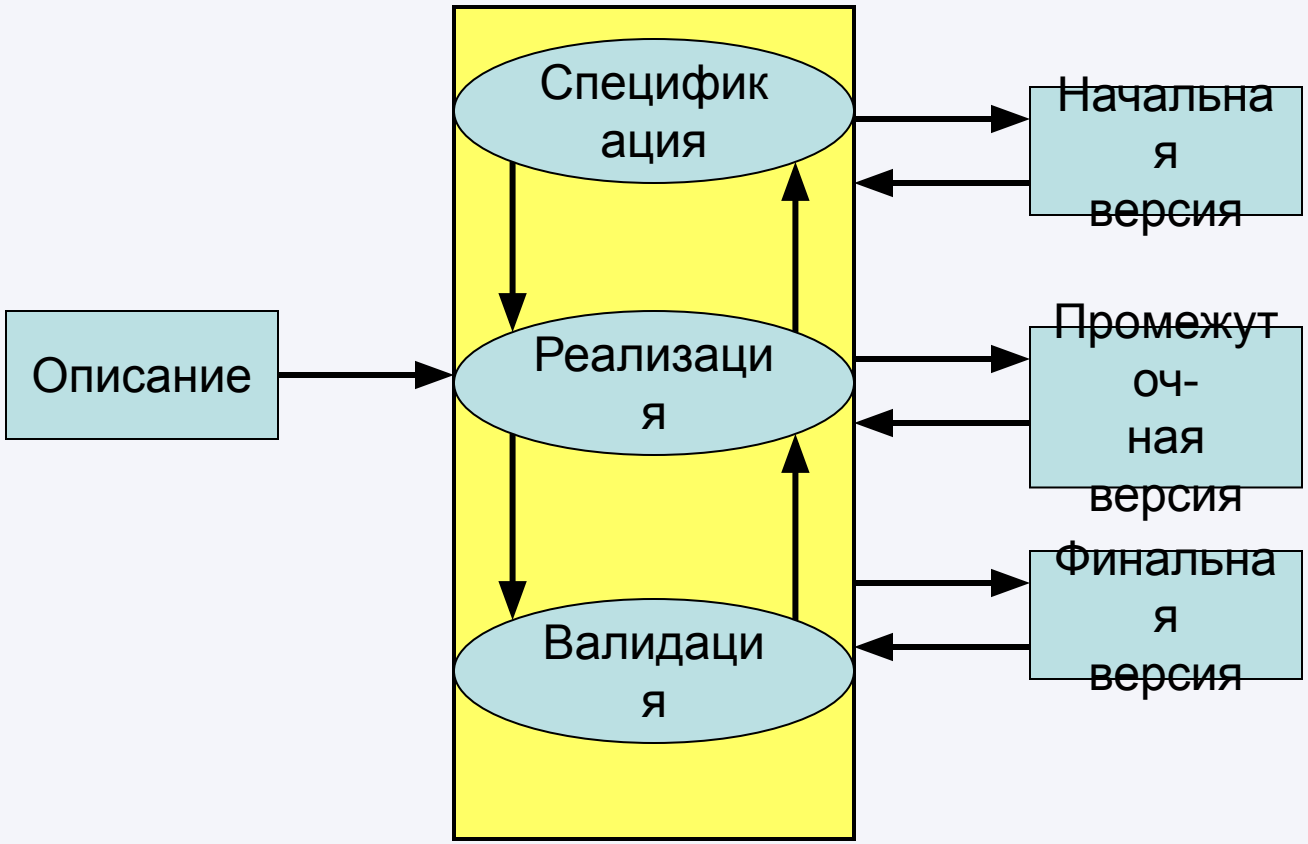
Прототипирование (3)

- Важно показать формы ввода данных, операторы, отчеты и интерактивные диалоги с пользователями
- Позволяет лучше понимать пользовательские потребности
- Помогает критически исследовать технические проблемы, связанные с разработкой продукта
- Обычно показывает крайние функциональные возможности, имеет низкую надежность, неэффективную реализацию

Эволюционная модель (1)

- В эволюционной модели проект реализуется последовательно, запускаясь с реализации подмножества UC и последовательно добавляя новый UC или их группы.
- Цель модели – работа с клиентом от начальной спецификации до заключительного SS
- Понимание требований SS увеличивается во время процесса. Начать с четкого и важного UC будет целеустремленно

Эволюционная модель (2)



Преимущества эволюционной модели

1. Удовлетворяет нашей "взъерошенной" жизни.
2. Она не требует высококвалифицированных системных аналитиков, работа которых оценивается очень дорого.
   - Она эффективна для больших SS, которые слишком сложны, чтобы охватывать.
   - Этот метод легко внедряется, проходит через жизненные этапы SS.

Недостатки эволюционной модели

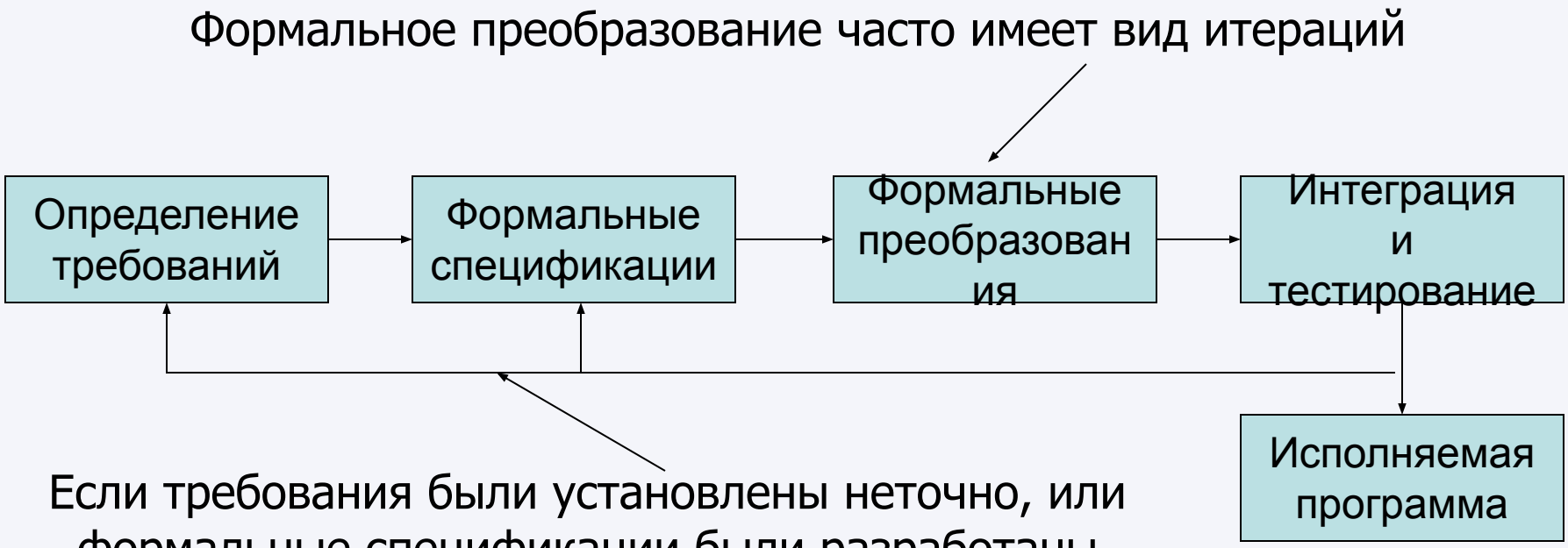
1. "Взъерошенный" процесс.
2. Заключительный SS плохо структурирован.
3. Не всегда возможно убедить пользователя работать с не полностью разработанным SS.

- Очень важна коммуникация разработчика с пользователем,нужна хороша психологическая среда.

Формальная модель (автоматизированный синтез SW)

- SS описывается математически и с поддержкой формальных методов, используя автоматические инструменты генерации кода. Окончательная версия SS сгенерирована итеративным способом (программа выполняется),
- Возможно генерировать только исполняемую программу, но документация, справочные файлы, т.е. все то, что нужно подготовить, часто является еще более трудным,
- Зависит от автоматического инструмента генерации кода (очень умный компилятор)

Разработка формальных систем



Преимущества формальной модели

1. Прежде всего необходимо понять очень хорошо требования SS и характеристики его работы.
2. Последующий процесс не является очень трудоемким
3. Довольно удобный реинжиниринг, быстрый и более дешевый.

- Это является подходящим для маленького SS и SS, которые имеют высокие требования к надежности; когда требования меняются (большая вероятность того, что требования будут часто менятьмя), когда жизнь приложения будет недолгой, система будет часто изменяться

Недостатки формальной модели

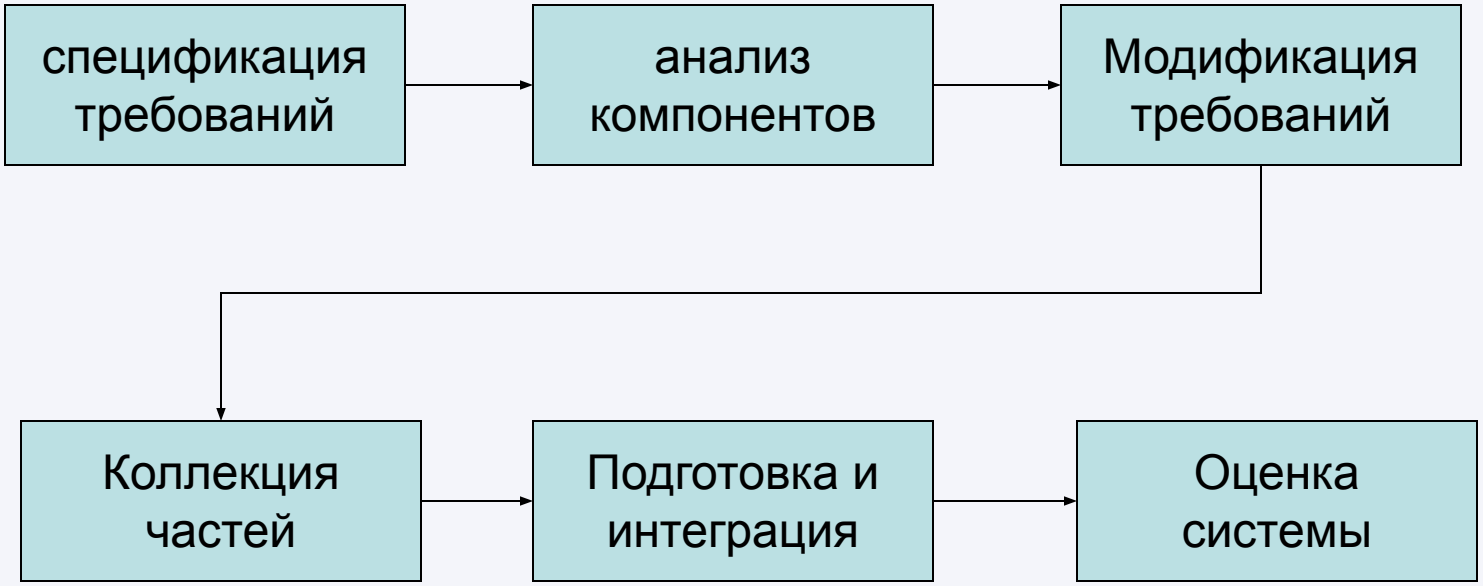
1. Процесс очень формален и сложен
2. Трудно описать системы формальном виде
3. Нужны специалисти высокой квалификации
4. Невозможно получить полноценную систему в способе формальных преобразований. Ручная работа также требуется, особенно разрабатывая интерфейс "человеческая система"

Программная модель многократного использования (1)

компонентная модель

- В мире есть много продуктов. Трудно обнаружить что-то новое. Остается находить то, что подходит Вам лучше всего и использовать это...
- Цель модели – использование существующих систем, подсистем или программ для того, чтобы решить возникшие задачи
- Новая задача – чтобы интегрировать части, разработанные различными разработчиками, в одну систему
- *Как это скучно...*

Программная модель многократного использования (2)



Преимущества программной модели многократного использования

1. Используется лучший продукт, разработанный другими. Возможно, что они знают лучше более узкий контекст приложения
2. Трудовые расходы ниже, это разрабатывается быстрее и более дешевое
3. Проще выдержать в тип продвижения

- *Это - модель будущего*

Недостатки программной модели многократного использования

1. Необходимо знать рынок продуктов очень хорошо
2. Нуждается в специалистах, которые в состоянии быть мастерами чужих продуктов в самом высоком уровне, часто без исходных текстов
3. Сложное взаимодействие между подсистемами, не всегда эффективные интерфейсы между подсистемами, различными рабочими средами с подсистемами

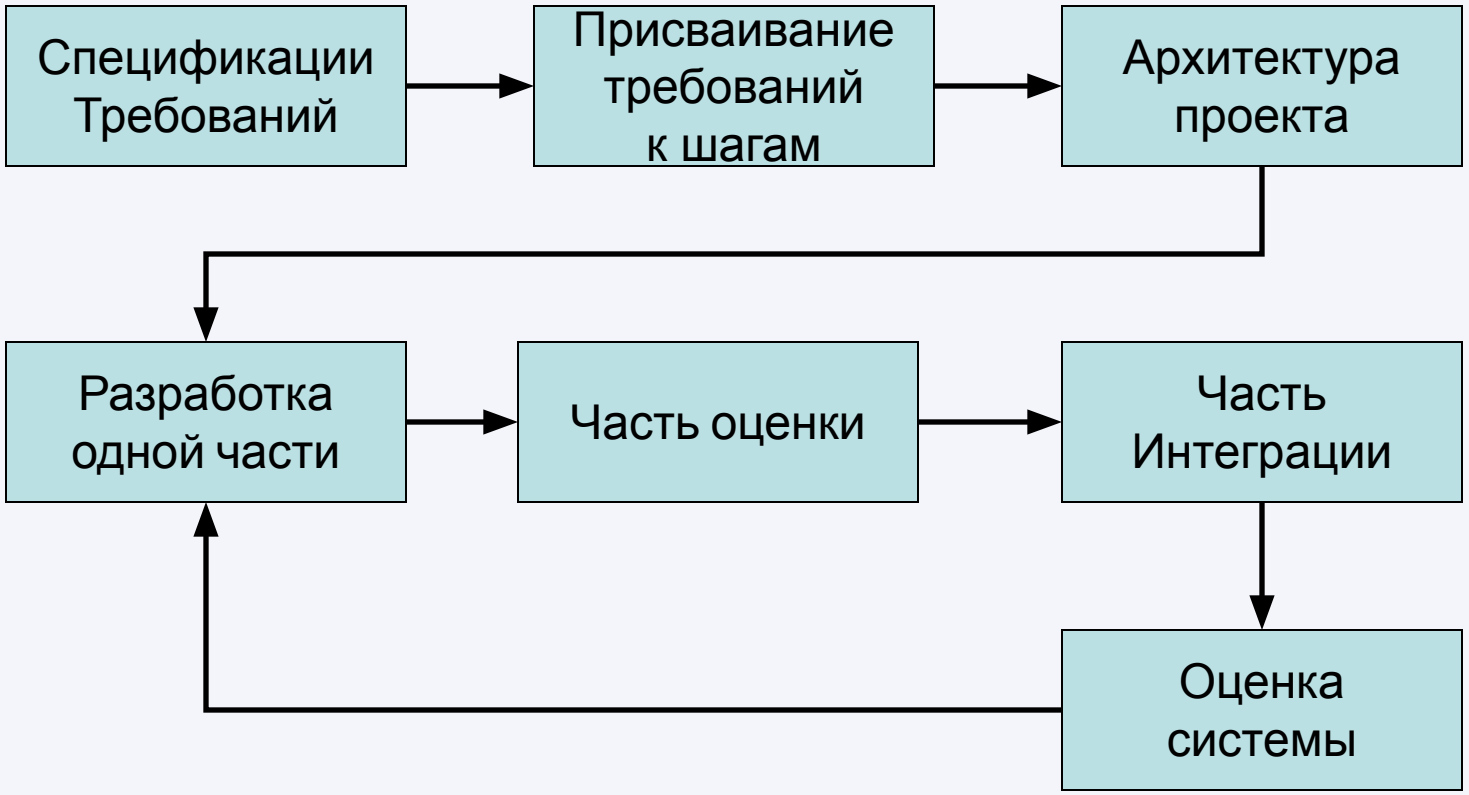
Итерации процесса

- Ресурсы итераций процесса:
  1. Некоторые модели процесса
  2. Жизненные циклы SS
  3. Изменение требований
  4. Ошибки
  5. ...
- Есть две модели итеративной организации: инкрементное и спиральная разработка SS

Поэтапная разработка (1)

1. Разработка системы разделена на несколько шагов, каждый раз, реализуя часть функциональных возможностей SS
2. Приоритеты к требованиям пользователя присваиваются, прежде всего анализируются UC самого высокого приоритета
3. Когда процесс запускается, разработчики пытаются не изменить требования, хотя это не всегда успешно

Поэтапная разработка (2)



Преимущества поэтапной разработки

1. Система может быть запущена, хотя полная функциональность еще не реализуется
2. В поэтапной разработке получается опыт, который может использоваться в других шагах
3. Процесс может быть скорректирован, риск отказа уменьшается
4. Разложение задачи облегчает реализацию

Недостатки поэтапной разработки

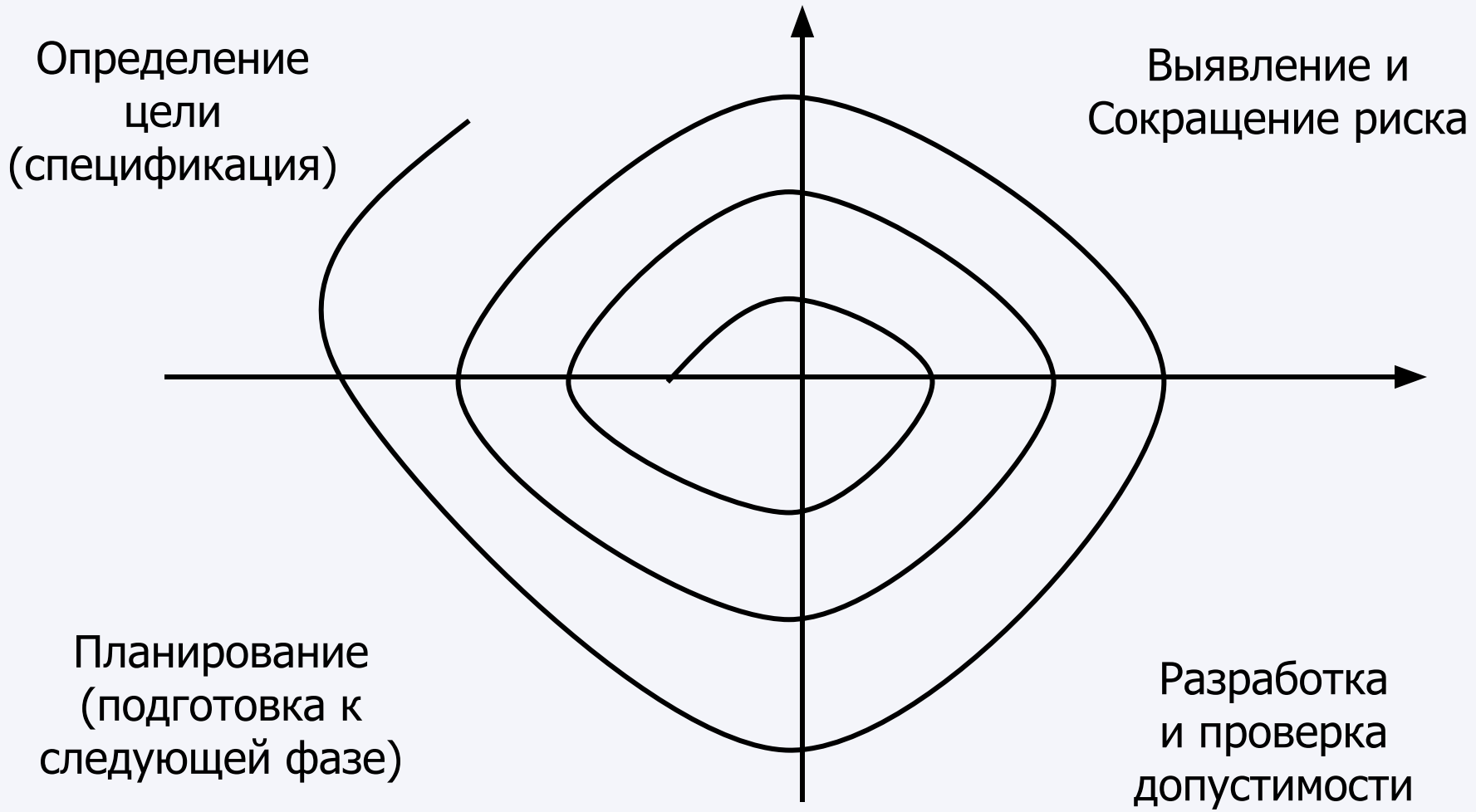
1. Проблема – интеграция отдельных частей
2. Возможно, что будет необходимо изменить архитектуру, когда много UC будут реализованы
3. Возможно, что SS не будет оптимален

- *Однако, много разработчиков делают так*

Спиральная модель

1. Процесс реализуется как спираль, а не как последовательность шагов с возвратами. Все, что уже разрабатывается, не перестраивается, но только улучшается
2. Нет никаких фиксированных фаз. Цели выбираются согласно потребностям
3. Риск высок. Здесь особенно оценивается и пробуется уменьшать риск.

Спиральная разработка



Спасибо за Ваше внимание!