1.2. Модели жизненного цикла разработки ПО

Жизненный цикл ПО. Структура жизненного цикла ПО.

Жизненный цикл (ЖЦ) ПО - это непрерывный процесс, который начинается с момента принятия решения о необходимости его создания и заканчивается в момент его полного изъятия из эксплуатации.

Основной нормативный документ, регламентирующий ЖЦ ПО - международный стандарт ISO/IEC 12207 (ISO - International Organization of Standardization - Международная организация по стандартизации, IEC - International Electrotechnical Commission - Международная комиссия по электротехнике). Он определяет структуру ЖЦ, содержащую процессы, действия и задачи, которые должны быть выполнены во время создания ПО.

Структура ЖЦ ПО (продолжение)

Структура ЖЦ ПО по стандарту ISO/IEC 12207 базируется на трех группах процессов:

1. **Основные процессы ЖЦ ПО**:
   - Приобретение;
   - Поставка;
   - Разработка;
   - Эксплуатация;
   - Сопровождение.

Структура ЖЦ ПО (продолжение)

2. Вспомогательные процессы, обеспечивающие выполнение основных процессов:

- Поддержка ПО;
- Документирование;
- Управление конфигурациями;
- Обеспечение качества;
- Верификация;
- Валидация;
- Совместные экспертизы;
- Аудит;
- Разрешение проблем.

Структура ЖЦ ПО (продолжение)

3. Организационные процессы:

- Управление проектами;
- Создание и управление инфраструктурой проекта; Совершенствование процессов ЖЦ;
- Работа с персоналом и его обучение.

Структура ЖЦ ПО (продолжение)

Процессы строятся из отдельных видов деятельности (activities). Например:

Приобретение ПО включает такие деятельности, как инициация приобретения, подготовка запроса предложений, подготовка контракта, анализ поставщиков, получение ПО и завершение приобретения.

Разработка ПО включает развертывание процесса разработки, анализ системных требований, проектирование программно-аппаратной системы в целом, анализ требований к ПО, проектирование архитектуры ПО, детальное проектирование, кодирование и отладочное тестирование, интеграцию ПО, квалификационное тестирование ПО, системную интеграцию, квалификационное тестирование системы, развертывание (установку или инсталляцию) ПО, поддержку процесса получения ПО.

Структура ЖЦ ПО (продолжение)

Эксплуатация включает в себя работы по внедрению компонентов ПО в эксплуатацию, в том числе конфигурирование, обеспечение эксплуатационной документацией, проведение обучения персонала и т.д., и непосредственно эксплуатацию/использование.

Сопровождение включает локализацию проблем и устранение причин их возникновения, модификацию ПО в рамках установленного регламента, подготовку предложений по совершенствованию, развитию и модернизации системы.

Управление проектом связано с вопросами планирования и организации работ, создания коллектива разработчиков и контроля за сроками и качеством выполняемых работ.

Структура ЖЦ ПО (продолжение)

Управление конфигурацией является одним из вспомогательных процессов, поддерживающих основные процессы жизненного цикла ПО, прежде всего процессы разработки и сопровождения ПО. При создании проектов сложных ИС, состоящих из многих компонентов, каждый из которых может иметь разновидности или версии, возникает проблема учета их связей и функций, создания унифицированной структуры и обеспечения развития всей системы. Управление конфигурацией позволяет организовать, систематически учитывать и контролировать внесение изменений в ПО на всех стадиях ЖЦ. Общие принципы и рекомендации конфигурационного учета, планирования и управления конфигурациями ПО отражены в проекте стандарта ISO 12207-2.

Структура ЖЦ ПО (продолжение)

Каждый вид деятельности нацелен на решение одной или нескольких задач (tasks). Например:

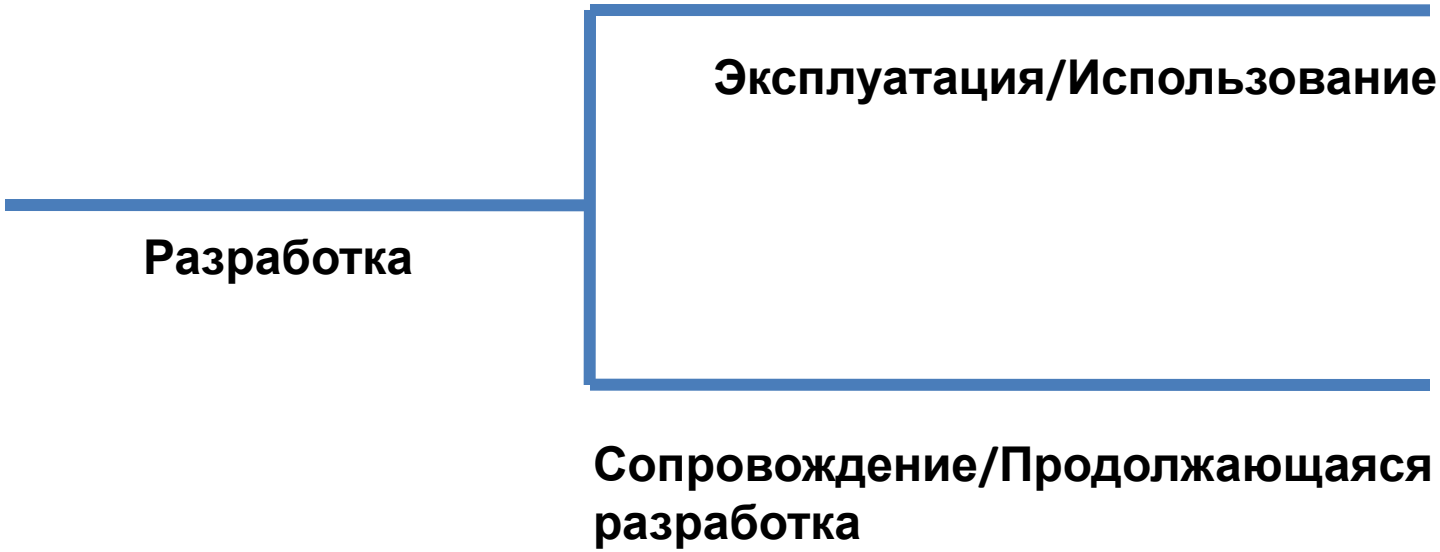
Развертывание процесса разработки состоит из определения модели жизненного цикла, документирования и контроля результатов отдельных работ, выбора используемых стандартов, языков, инструментов и пр.

Структура ЖЦ ПО (продолжение)

Каждый процесс или подпроцесс характеризуется определенными задачами и методами их решения, исходными данными, полученными на предыдущем этапе, и результатами.

ЖЦ ПО носит итерационный характер: результаты очередного этапа часто вызывают изменения в проектных решениях, выработанных на более ранних этапах.

Структура ЖЦ ПО (продолжение)



Основные характеристики фазы использования, влияющие на разработку ПО

1. Периодичность использования.
2. Количество пользователей.
3. Ограничения по времени выполнения запросов и точности выходной информации.
4. Переносимость на другие платформы. Требования к операционной системе и техническим средствам.
5. Возможность интеграции с другими программами.
6. Последствия отказов.

Модели жизненного цикла разработки ПО

Модели жизненного цикла разработки ПО

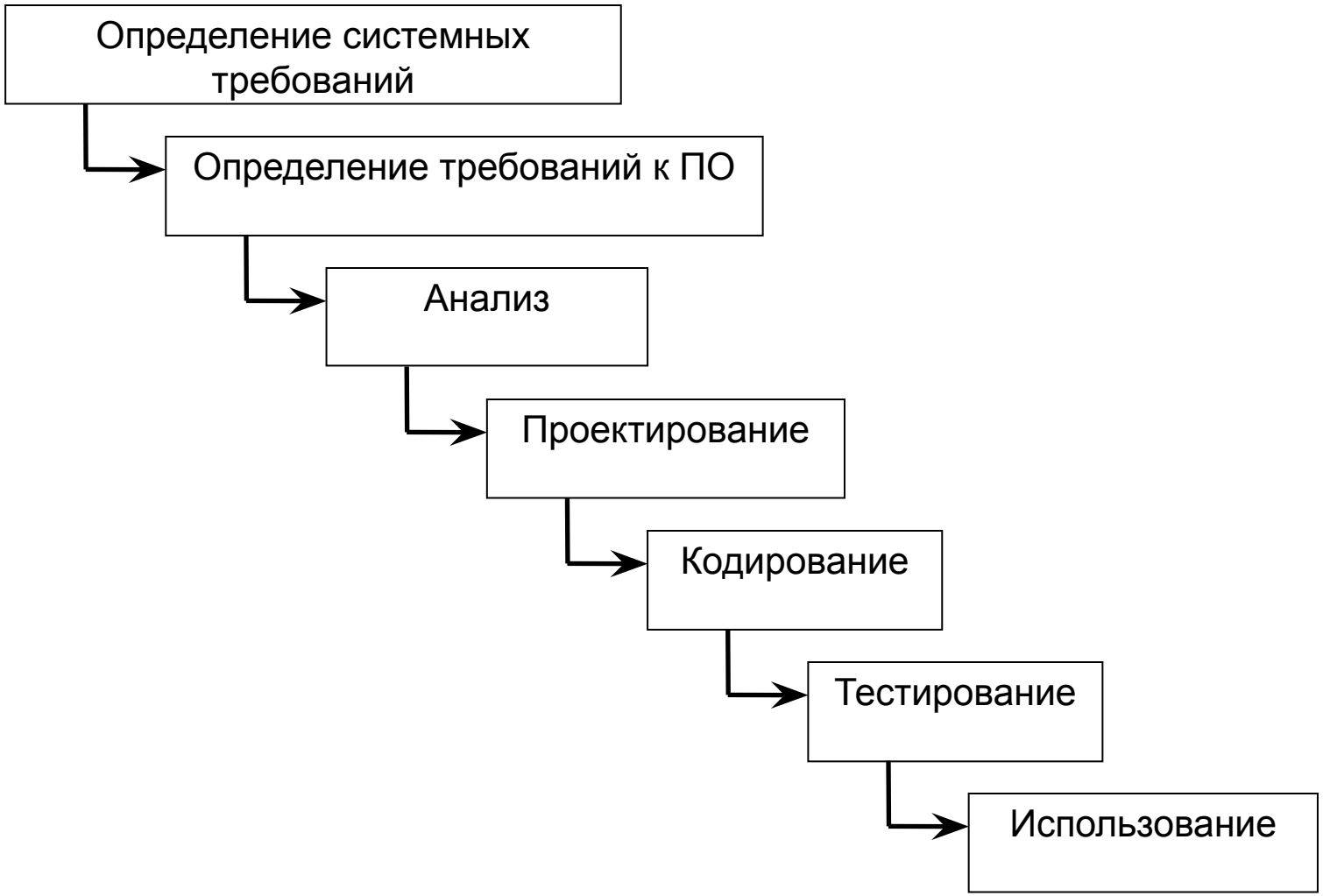
- Стандарт ISO/IEC 12207 не предлагает конкретную модель ЖЦ и методы разработки ПО (под моделью ЖЦ понимается структура, определяющая последователь-ность выполнения и взаимосвязи процессов, действий и задач, выполняемых на протяжении ЖЦ).
- Модель ЖЦ зависит от специфики ПО и специфики условий, в которых последнее создается и функционирует.
- Стандарт ISO/IEC 12207 описывает структуру процессов ЖЦ ПО, но не конкретизирует в деталях, как реализовать или выполнить действия и задачи, включенные в эти процессы. Его регламенты являются общими для любых моделей ЖЦ, методологий и технологий разработки.

Модели жизненного цикла разработки ПО

Основные модели ЖЦ:

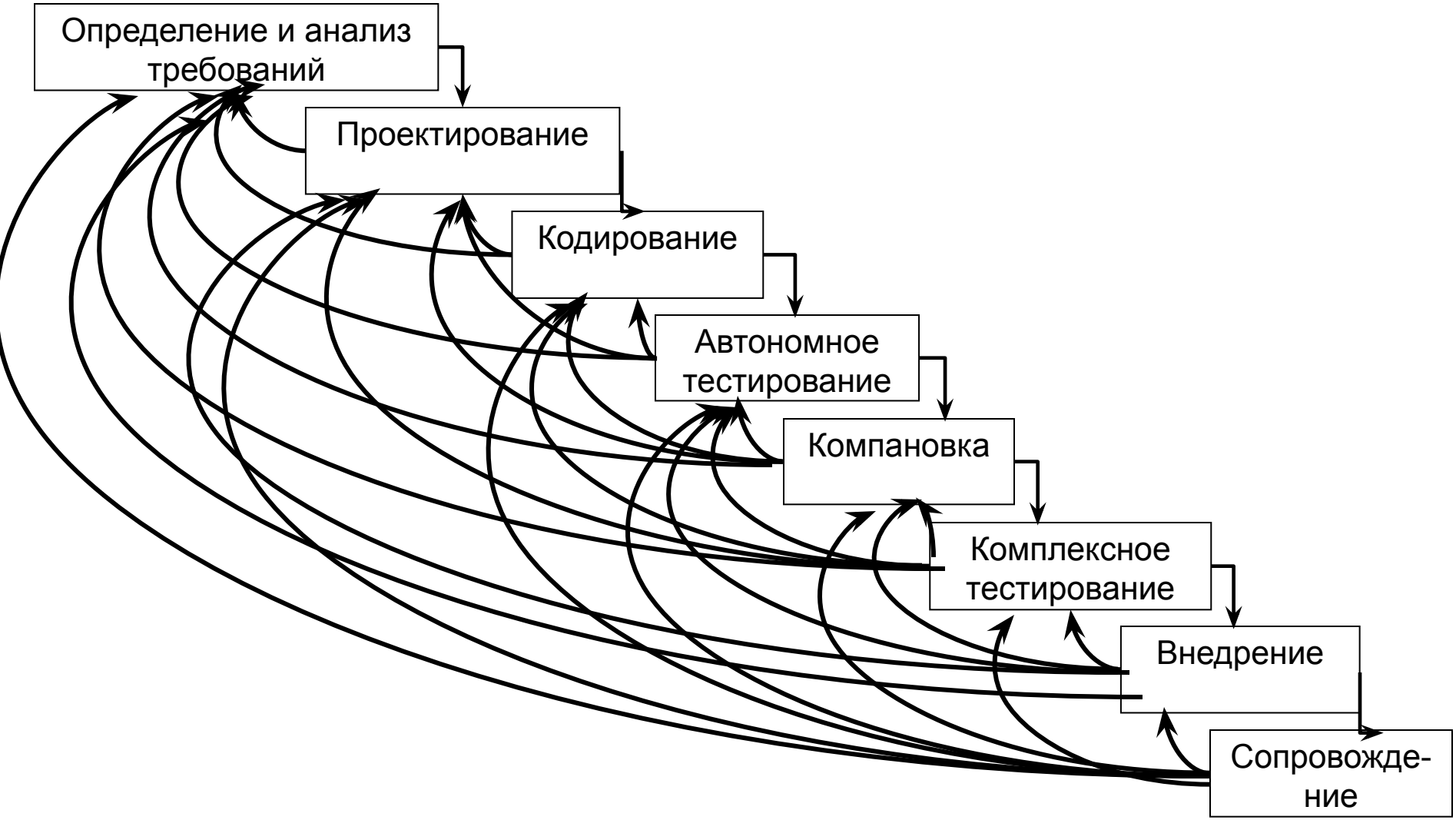
- Водопадная (каскадная модель) (70-85 г.г.);
- Спиральная модель (86-90 г.г.).

Классический жизненный цикл



Последовательность разработки согласно «общепринятой» водопадной модели

Классический жизненный цикл



Реальный процесс разработки по водопадной модели

Классический жизненный цикл

Достоинства классического жизненного цикла:

дает план и временной график по всем этапам проекта, упорядочивает ход разработки.

Классический жизненный цикл

Недостатки классического жизненного цикла:

- существенное запаздывание с получением результатов.
- реальные проекты часто требуют отклонения от стандартной последовательности шагов;
- цикл основан на точной формулировке исходных требований к ПО (реально в начале проекта требования заказчика определены лишь частично).
- Модели (как функциональные, так и информационные) автоматизируемого объекта могут устареть одновременно с их утверждением;
- результаты проекта доступны заказчику только в конце работы. Таким образом, пользователи могут внести свои замечания только после того, как работа над системой будет полностью завершена.

Стратегии разработки ПО

- ***однократный проход*** (водопадная стратегия) — линейная последовательность этапов конструирования;
- ***итеративная стратегия.*** В начале процесса определяются все пользовательские и системные требования, оставшаяся часть конструирования выполняется в виде последовательности версий. Первая версия реализует часть запланированных возможностей, следующая версия реализует дополнительные возможности и т. д., пока не будет получена полная система;
- ***эволюционная стратегия.*** Система также строится в виде последовательности версий, но в начале процесса определены не все требования. Требования уточняются в результате разработки версий.

Стратегии разработки ПО

Характеристики стратегий конструирования ПО в соответствии с требованиями стандарта IEEE/EIA 12207.2.

Стратегия конструирования	В начале процесса определены все требования?	Множество циклов разработки?	Промежуточное ПО распространяется?
Однократный проход	Да	Нет	Нет
Итеративная (запланированное улучшение продукта)	Да	Да	Может быть
Эволюционная	Нет	Да	Да

Макетирование

Макетирование (прототипирование) — это процесс создания модели требуемого программного продукта.

Основная цель макетирования — снять неопределенности в требованиях заказчика.

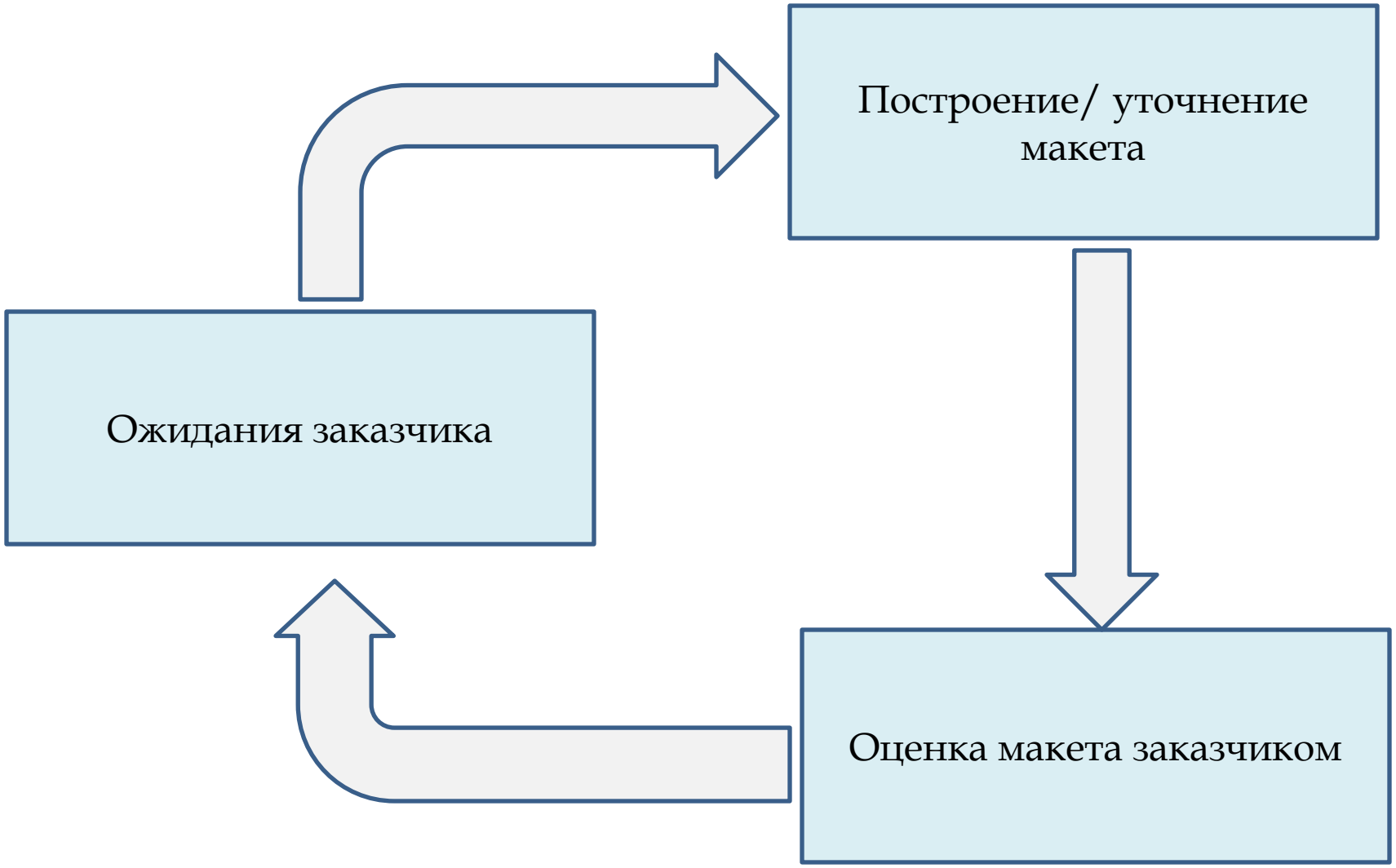
Модель может принимать одну из трех форм:

1) бумажный макет или макет на основе ПК (изображает или рисует человеко-машинный диалог);

2) работающий макет (выполняет некоторую часть требуемых функций);

3) существующая программа (характеристики которой затем должны быть улучшены).

Макетирование



Макетирование

Итерации повторяются до тех пор, пока макет не выявит все требования заказчика и, тем самым, не даст возможность разработчику понять, что должно быть сделано.

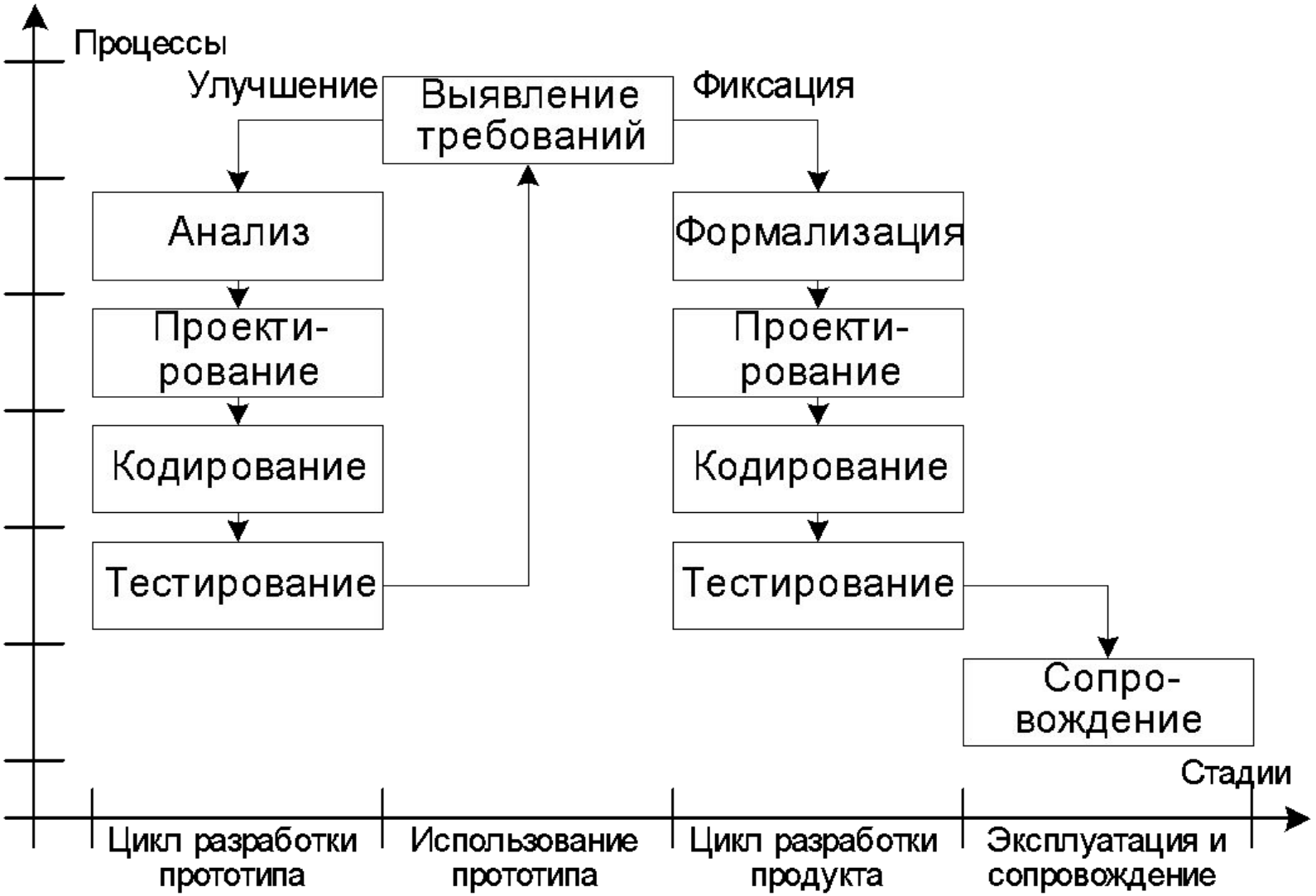
Достоинство макетирования: обеспечивает определение полных требований к ПО.

Недостатки макетирования:

- заказчик может принять макет за продукт;
- разработчик может принять макет за продукт.

Макетирование

Классическая модель прототипирования



Спиральная модель

Спиральная модель делает упор на начальные этапы ЖЦ: анализ и проектирование. На этих этапах реализуемость технических решений проверяется путем создания прототипов.

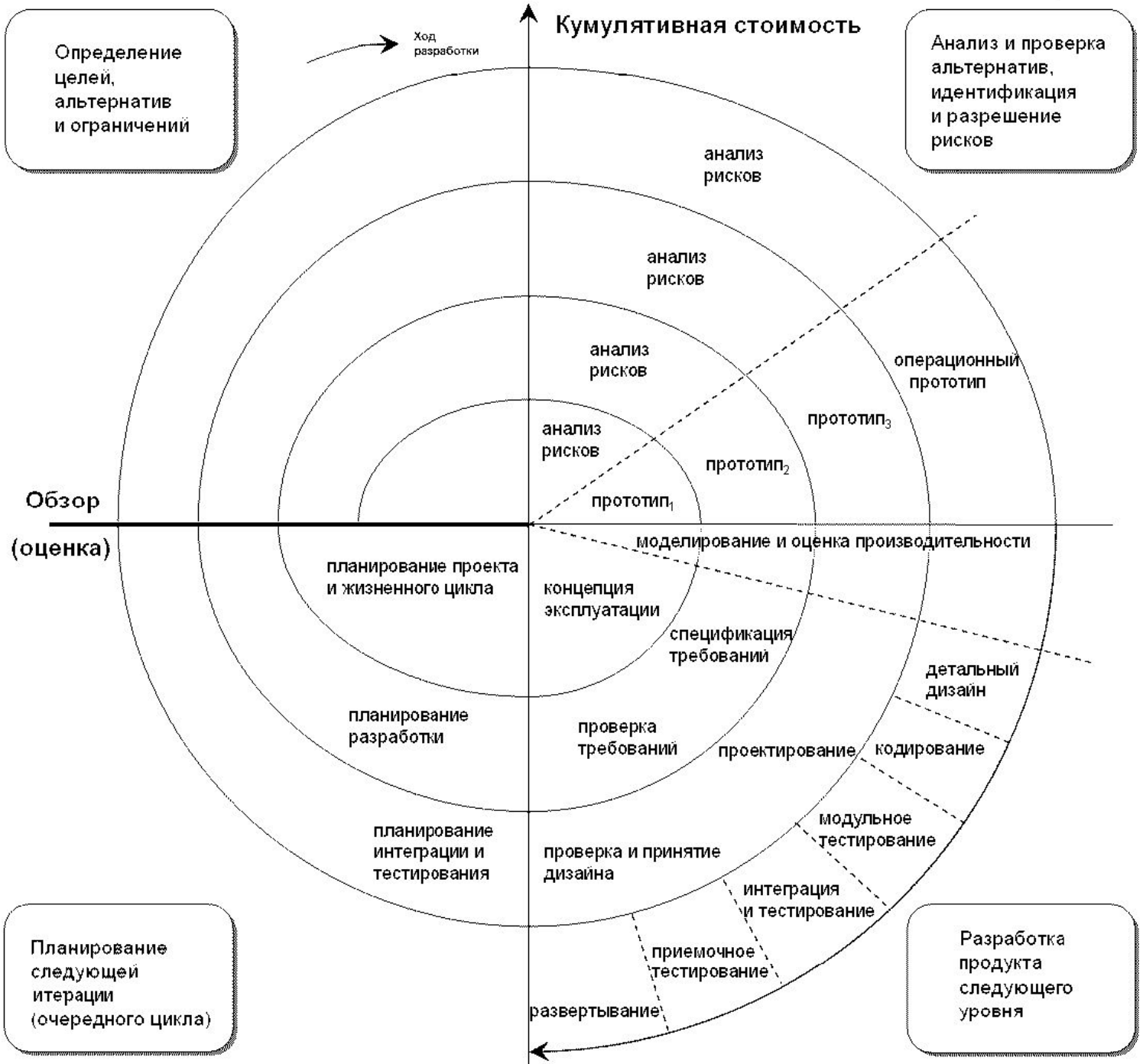
Каждый виток спирали соответствует созданию фрагмента или версии ПО, на нем уточняются цели и характеристики проекта, определяется его качество и планируются работы следующего витка спирали. В результате выбирается обоснованный вариант, который доводится до реализации.

Спиральная модель (автор Барри Боэм, 1988) базируется на лучших свойствах классического жизненного цикла и макетирования, к которым добавляется новый элемент — анализ риска, отсутствующий в этих парадигмах.

Спиральная модель

1 — начальный сбор требований и планирование проекта;
2 — та же работа, но на основе рекомендаций заказчика; 3 — анализ риска на основе начальных требований; *4* — анализ риска на основе реакции заказчика; 5 — переход к комплексной системе; 6 — начальный макет системы; *7* — следующий уровень макета; *8* — сконструированная система; 9 — оценивание заказчиком

Спиральная модель



Спиральная модель

Достоинства спиральной модели:

1) наиболее реально (в виде эволюции) отображает разработку программного обеспечения;

2) позволяет явно учитывать риск на каждом витке эволюции разработки;

3) включает шаг системного подхода в итерационную структуру разработки;

4) использует моделирование для уменьшения риска и совершенствования программного изделия.

Спиральная модель

Недостатки спиральной модели:

1. Трудность приведения проекта к конечному результату;
2. Сложность планирования (определения количества и длительности итераций, оценки затрат и рисков)
3. Сложность применения модели с точки зрения менеджера и заказчиков (из – за привычки к строгому и детальному планированию);
4. Повышенные требования к заказчику.