ЖЦ ПО:

Модели жизненного цикла программного обеспечения

Блок 1: ЖЦ ПО

ЖЦ ПО

Стратегия жизненного цикла программного обеспечения – порядок следования и содержания основных этапов процесса разработки.

Модель жизненного цикла программного обеспечения – структура, содержащая процессы действия и задачи, которые осуществляются в ходе разработки, использования и сопровождения программного продукта.

Жизненный цикл программного обеспечения определяется как период времени, который начинается с момента принятия решения о необходимости создания ПО и заканчивается в момент его полного изъятия из эксплуатации.

ЖЦ ПО

Основным нормативным документом, регламентирующим состав процессов ЖЦ ПО, является ***международный стандарт ISO/IEC 12207:1995 "Информационные технологии - Процессы жизненного цикла программного обеспечения " (ISO - International Organization for 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.

ЖЦ ПО: понятия

В данном стандарте **ПО** (или программный продукт) определяется как набор компьютерных программ, процедур и, возможно, связанной с ними документации и данных.

Процесс определяется как совокупность взаимосвязанных действий, преобразующих некоторые входные данные в выходные.

Каждый процесс ***характеризуется определенными задачами и методами их решения, исходными данными, полученными от других процессов, и результатами***.

Каждый процесс разделен на набор действий, каждое действие - на набор задач. Каждый процесс, действие или задача инициируется и выполняется другим процессом по мере необходимости, причем не существует заранее определенных последовательностей выполнения (естественно, при сохранении связей по входным данным).

Процессы ЖЦ ПО

В соответствии со стандартом ISO/IEC 12207 все процессы ЖЦ ПО разделены на три группы процессов:

1) **основные процессы ЖЦ ПО** (приобретение, поставка, разработка, эксплуатация, сопровождение);
2) **вспомогательные процессы** (документирование, управление конфигурацией, обеспечение качества, верификация, аттестация, оценка, аудит, решение проблем);
3) **организационные процессы** (управление проектами, создание инфраструктуры проекта, определение, оценка и улучшение самого ЖЦ ПО, обучение).

Основные процессы ЖЦ ПО

- **Процесс приобретения (acquisition process**). Он состоит из действий и задач заказчика, приобретающего ПО.
- **Процесс поставки (supply process).** Он охватывает действия и задачи, выполняемые поставщиком, который снабжает заказчика программным продуктом или услугой.
- **Процесс разработки (development process**). Он предусматривает действия и задачи, выполняемые разработчиком, и охватывает работы по созданию ПО и его компонентов в соответствии с заданными требованиями, включая оформление проектной и эксплуатационной документации, подготовку материалов, необходимых для проверки работоспособности и соответствующего качества программных продуктов, материалов, необходимых для организации обучения персонала, и т. д.
- **Процесс эксплуатации (operation process**). Он охватывает действия и задачи оператора — организации, эксплуатирующей систему.
- **Процесс сопровождения (maintenance process).** Он предусматривает действия и задачи, выполняемые сопровождающей организацией (службой сопровождения).

Вспомогательные процессы ЖЦ ПО

- **Процесс документирования (documentation process**). Он предусматривает формализованное описание информации, созданной в течение ЖЦ ПО. Данный процесс состоит из набора действий, с помощью которых планируют, проектируют, разрабатывают, выпускают, редактируют, распространяют и сопровождают документы, необходимые для всех заинтересованных лиц, таких, как руководство, технические специалисты и пользователи системы.
- **Процесс управления конфигурацией (configuration management process**). Он предполагает применение административных и технических процедур на всем протяжении ЖЦ ПО для определения состояния компонентов ПО в системе, управления модификациями ПО, описания и подготовки отчетов о состоянии компонентов ПО и запросов на модификацию, обеспечения полноты, совместимости и корректности компонентов ПО, управления хранением и поставкой ПО.
- **Процесс обеспечения качества (quality assurance process**). Он обеспечивает соответствующие гарантии того, что ПО и процессы его ЖЦ соответствуют заданным требованиям и утвержденным планам. Под качеством ПО понимается совокупность свойств, которые характеризуют способность ПО удовлетворять заданным требованиям.

Вспомогательные процессы ЖЦ ПО

- **Процесс верификации (verification process).** Он состоит в определении того, что программные продукты, являющиеся результатами некоторого действия, полностью удовлетворяют требованиям или условиям, обусловленным предшествующими действиями (верификация в узком смысле означает формальное доказательство правильности ПО).
- **Процесс аттестации (validation process).** Он предусматривает определение полноты соответствия заданных требований и созданной системы или программного продукта их конкретному функциональному назначению. Под аттестацией обычно понимается подтверждение и оценка достоверности проведенного тестирования ПО.
- **Процесс совместной оценки (joint review process).** Он предназначен для оценки состояния работ по проекту и ПО, создаваемого при выполнении данных работ (действий). Он сосредоточен в основном на контроле планирования и управления ресурсами, персоналом, аппаратурой и инструментальными средствами проекта.
- **Процесс аудита (audit process).** Он представляет собой определение соответствия требованиям, планам и условиям договора. Аудит может выполняться двумя любыми сторонами, участвующими в договоре, когда одна сторона проверяет другую. Аудит — это ревизия (проверка), проводимая компетентным органом (лицом) в целях обеспечения независимой оценки степени соответствия ПО или процессов установленным требованиям.
- **Процесс разрешения проблем (problem resolution process**). Он предусматривает анализ и решение проблем (включая обнаруженные несоответствия) независимо от их происхождения или источника, которые обнаружены в ходе разработки, эксплуатации, сопровождения или других процессов. Каждая обнаруженная проблема должна быть идентифицирована, описана, проанализирована и разрешена.

Организационные процессы ЖЦ ПО

- **Процесс управления (management process).** Он состоит из действий и задач, которые могут выполняться любой стороной, управляющей своими процессами. Данная сторона (менеджер) отвечает за управление выпуском продукта, управление проектом и управление задачами соответствующих процессов, таких, как приобретение, поставка, разработка, эксплуатация, сопровождение и др.
- **Процесс создания инфраструктуры (infrastructure process).** Он охватывает выбор и поддержку (сопровождение) технологии, стандартов и инструментальных средств, выбор и установку аппаратных и программных средств, используемых для разработки, эксплуатации или сопровождения ПО.
- **Процесс усовершенствования (improvement process).** Он предусматривает оценку, измерение, контроль и усовершенствование процессов ЖЦ ПО.
- **Процесс обучения (training process).** Он охватывает первоначальное обучение и последующее постоянное повышение квалификации персонала.

ГОСТ 19.701-90 ЕСПД. СХЕМЫ АЛГОРИТМОВ, ПРОГРАММ, ДАННЫХ И СИСТЕМ :

Настоящий стандарт распространяется на условные обозначения (символы) в схемах алгоритмов, программ, данных и систем и устанавливает правила выполнения схем, используемых для отображения различных видов задач обработки данных и средств их решения.

ГОСТ 19.401-78 ЕСПД. ТЕКСТ ПРОГРАММЫ. ТРЕБОВАНИЯ К СОДЕРЖАНИЮ И ОФОРМЛЕНИЮ :

Требования к оформлению текста программы достаточно просты и естественны для грамотного программиста. Основное, чем требуется руководствоваться при создании этого документа – это то, что текст программы должен быть удобочитаемым.

По-прежнему обязательным является составление информационной части - аннотации и содержания.

Основная часть документа должна состоять из текстов одного или нескольких разделов, которым даны наименования.

Текст каждого программного файла начинается с "шапки", в которой указывается:

- наименование программы,
- автор,
- дата создания программы,
- номер версии,
- дата последней модификации.
- Обязательными являются комментарии, а также строгое соблюдение правил отступа.

ГОСТ 19.402-78 ЕСПД. ОПИСАНИЕ ПРОГРАММЫ :

Этот стандарт ориентирован на документирование результирующего продукта разработки.

Описание программы обязательно должно включать информационную часть - аннотацию и содержание.

Блок 2:
Модели ЖЦ ПО

Модели жизненного цикла

Основные модели жизненного цикла ПО:

- Каскадная модель
- Макетирование
- Инкрементная модель
- Спиральная модель

Виды стратегий жизненного цикла

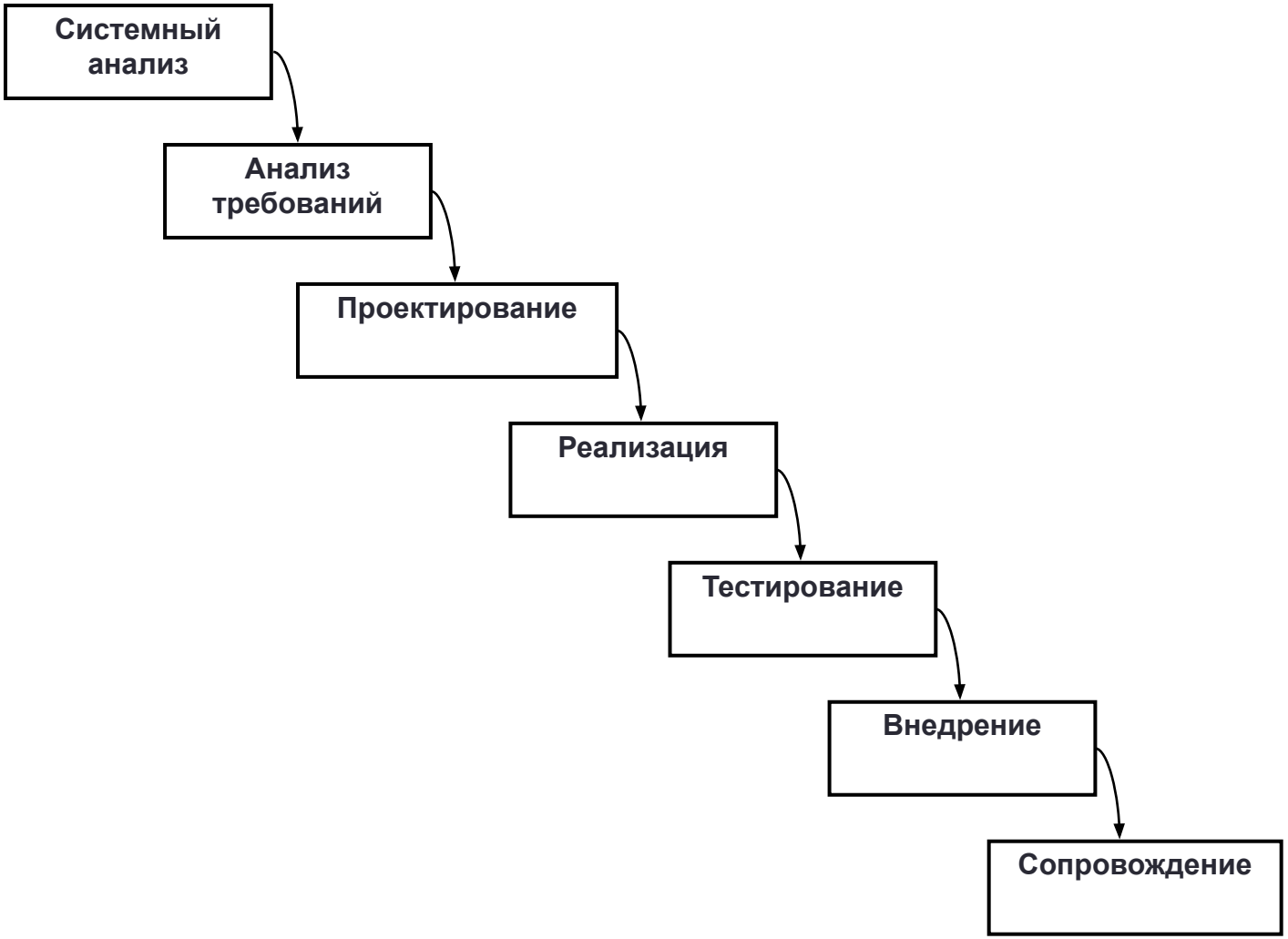
- ***однократный проход*** (водопадная стратегия) – линейная последовательность этапов конструирования;
- ***инкрементная стратегия*** – разработка ПО ведется с определения всех требований к ПО, а оставшаяся часть разработки выполняется в виде последовательности версий;
- ***эволюционная стратегия*** – ПО также разрабатывается в виде версий, но требования формируются и уточняются по ходу разработки от версии к версии

Виды стратегий жизненного цикла

Стратегия конструирования	Определение требований в начале разработки	Количество циклов разработки	Распространение промежуточного ПО
Однократный проход	Все	Один	**Нет**
Инкрементная стратегия	Все	Несколько	**Возможно**
Эволюционная стратегия	**Только часть требований**	**Несколько**	**Да**

Каскадная (водопадная) модель

Автор: Уинстон Ройс, 1970 г



Каскадная (водопадная) модель

Этап системного анализа:

- задается роль каждого элемента в системе и их взаимодействие друг с другом;
- формируется интерфейс ПО с другими элементами (аппаратурой, базами данных, пользователями).
- начинается решение задачи планирования проекта.

Этап анализа требований:

- детальное определение функциональных и нефункциональных требований, предъявляемых к разрабатываемому ПО;
- завершается задача планирования проекта.

Каскадная (водопадная) модель

Этап проектирования:

- создание представлений:
  - архитектуры ПО;
  - модульной структуры ПО;
  - алгоритмической структуры ПО;
  - структуры данных;
  - графического интерфейса пользователя.

- оценка качества будущего программного обеспечения.

Каскадная (водопадная) модель

Этап реализации:

- преобразование проектных спецификаций в текст на языке программирования (кодирование).

Этап тестирования:

- проверка корректности выполнения программы;
- обнаружение и исправление ошибок в функциях, логике и форме реализации программного продукта.

Этап внедрения:

- выполнение установки разработанного ПО у заказчика;
- обучение персонала;
- плавный переход от старого ПО (если оно есть) к использованию нового.

Каскадная (водопадная) модель

Этап сопровождения:

Внесение изменений в эксплуатируемое ПО с целями:

- исправления ошибок;
- адаптации к изменениям внешней для ПО среды;
- усовершенствования ПО по требованиям заказчика.

Преимущества каскадной модели

- широкая известность и простота модели;
- упорядоченность преодоления сложностей и хорошо срабатывает для тех проектов, которые достаточно понятны, но все же трудно разрешимы;
- отличается стабильностью требований;
- удобна, когда требования к качеству доминируют над требованиями к затратам и графику выполнения проекта;
- способствует осуществлению строгого контроля менеджмента проекта;
- позволяет участникам проекта, завершившим действия на выполняемой ими фазе, принять участие в реализации других проектов;
- определяет процедуры по контролю за качеством;
- стадии модели довольно хорошо определены и понятны, легко отслеживаются с помощью временной шкалы или графика Гантта.

Недостатки каскадной модели

- в основе модели лежит последовательная линейная структура;
- невозможность предотвращения возникновение итераций между фазами;
- она может создать ошибочное впечатление о работе над проектом;
- интеграция всех полученных результатов происходит внезапно в завершающей стадии работы модели и у клиента практически нет возможности ознакомиться с системой заранее;
- каждая фаза является предпосылкой для выполнения последующих действий и ее результат считается замороженным;
- все требования должны быть известны в начале жизненного цикла;
- необходимость в жестком управлении и контроле;
- модель основана на документации;
- весь программный продукт разрабатывается за один раз;
- отсутствует возможность учесть переделку и итерации за рамками проекта.

Каскадная модель жизненного цикла

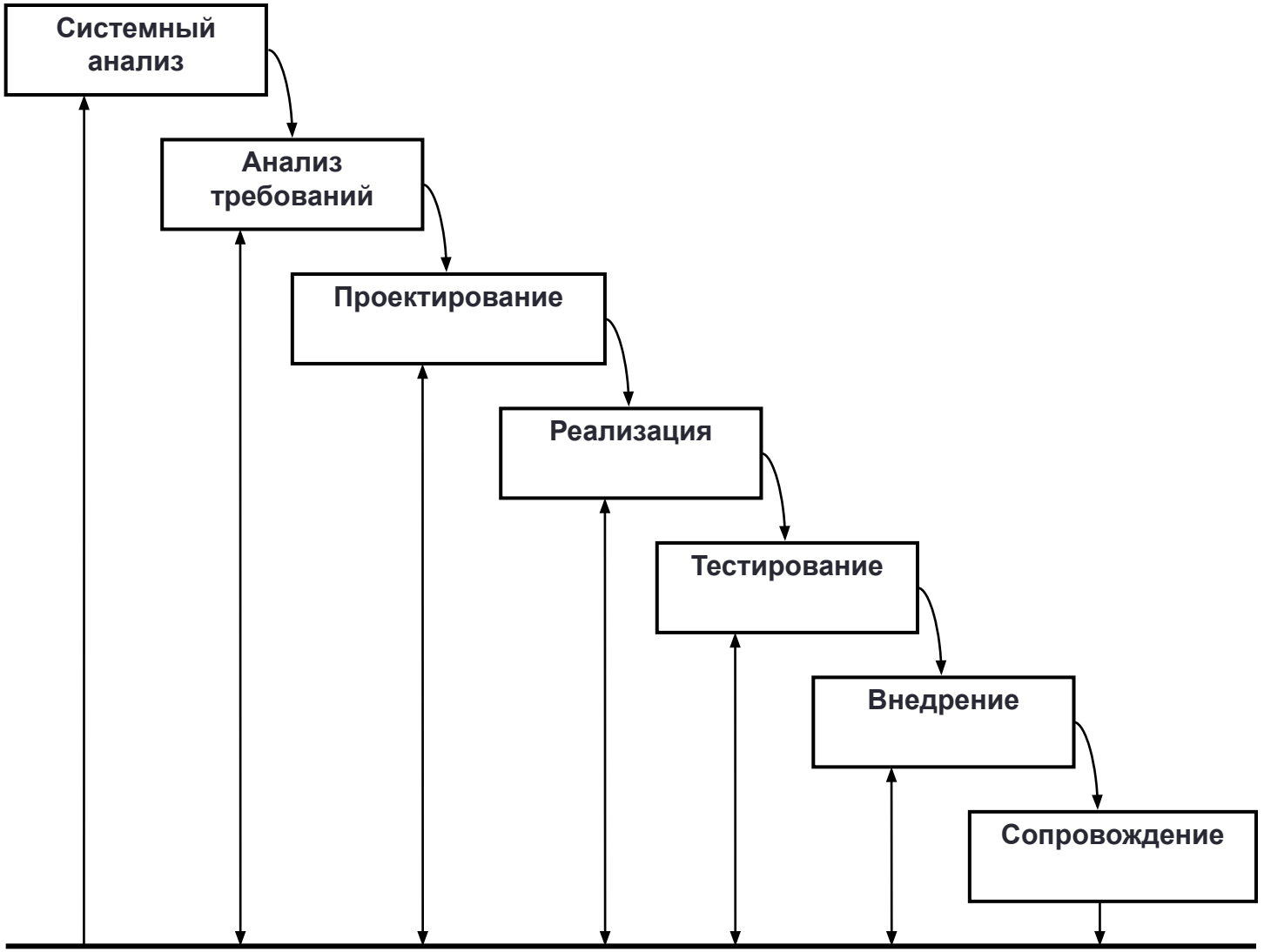
Критерии применения каскадной модели:

- требования к ПО и их реализация максимально четко определены и понятны;
- неизменяемое определение продукта и вполне понятные технические методики;
- если компания имеет опыт построения подобного рода систем.

Область применения:

- сложные системы с большим количеством задач вычислительного характера,
- системы управления производственными процессами повышенной опасности и др.

Модель водоворота



V-образная модель

V-образная модель

- **планирование проекта и требований** – определяются системные требования, а также то, каким образом будут распределены ресурсы организации с целью их соответствия поставленным требованиям;
- **анализ требований к продукту и его спецификации** – анализ существующей на данный момент проблемы с ПО, завершается полной спецификацией ожидаемой внешней линии поведения создаваемой программной системы;
- **архитектура или проектирование на высшем уровне** – определяет, каким образом функции ПО должны применяться при реализации проекта;
- **детализированная разработка проекта** – определяет и документально обосновывает алгоритмы для каждого компонента, который был определен на фазе построения архитектуры;

V-образная модель

- **планирование проекта и требований** – определяются системные требования, а также то, каким образом будут распределены ресурсы организации с целью их соответствия поставленным требованиям;
- **модульное тестирование** – выполняется проверка каждого закодированного модуля на наличие ошибок;
- **интеграция и тестирование** – установка взаимосвязей между группами ранее поэлементно испытанных модулей с целью подтверждения того, что эти группы работают также хорошо, как и модули, испытанные независимо друг от друга на этапе поэлементного тестирования;
- **системное и приемочное тестирование** – выполняется проверка функционирования программной системы в целом, после помещения в ее аппаратную среду в соответствии со спецификацией требований к ПО;

V-образная модель

- **производство, эксплуатация и сопровождение** – ПО запускается в производство;
- **приемочные испытания** – позволяет пользователю протестировать функциональные возможности системы на соответствие исходным требованиям.

Преимущества V-образной модели

- в модели особое значение придается планированию, направленному на верификацию и аттестацию разрабатываемого продукта на ранних стадиях его разработки;
- в модели предусмотрены аттестация и верификация всех внешних и внутренних полученных данных, а не только самого программного продукта;
- в V-образной модели определение требований выполняется перед разработкой проекта системы, а проектирование ПО – перед разработкой компонентов;
- модель определяет продукты, которые должны быть получены в результате процесса разработки;
- благодаря модели менеджеры проекта может отслеживать ход процесса разработки, так как в данном случае вполне возможно воспользоваться временной шкалой, а завершение каждой фазы является контрольной точкой;
- модель проста в использовании.

Недостатки V-образной модели

- с ее помощью непросто справиться с параллельными событиями;
- в ней не учтены итерации между фазами;
- в модели не предусмотрено внесение требования динамических изменений на разных этапах жизненного цикла;
- тестирование требований в жизненном цикле происходит слишком поздно, вследствие чего невозможно внести изменения, не повлияв при этом на график выполнения проекта;
- в модель не входят действия, направленные на анализ рисков.

Область применения V-образной модели

V-образная модель лучше всего срабатывает тогда, когда вся информация о требованиях доступна заранее.

Использование модели эффективно в том случае, когда доступными являются информация о методе реализации решения и технология, а персонал владеет необходимыми умениями и опытом в работе с данной технологией.

V-образная модель – это отличный выбор для систем, в которых требуется высокая надежность.

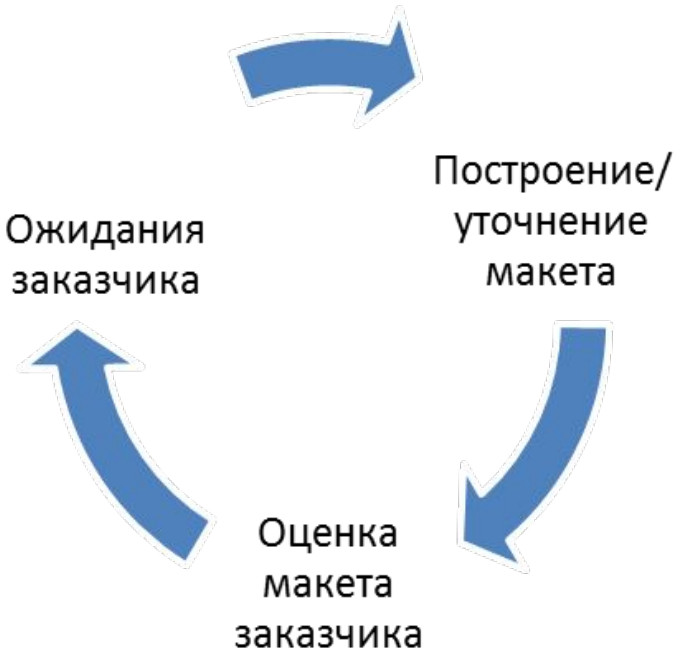
Макетирование

Макетирование (***прототипирование***) – это процесс создания модели разрабатываемого программного продукта.

Модель может принимать один из трех видов:

- бумажный макет или «электронный» макет, который представляет человеко-машинный интерфейс;
- работающий макет (выполняет только часть требуемых функций);
- существующая программа (характеристики которой должны быть улучшены).

Процесс макетирования



Преимущества макетирования

- конечный пользователь может "увидеть" системные требования в процессе их сбора командой разработчиков;
- снижается возможность возникновения путаницы, искажения информации или недоразумений при определении системных требований;
- возможность внесения новых или неожиданных требований пользователя;
- минимизация возникновения разногласий при общении заказчиков с разработчиками;
- модель позволяет выполнять гибкое проектирование и разработку;
- образуются постоянные, видимые признаки прогресса в выполнении проекта;

Преимущества макетирования

- принимая участие в процессе разработки на протяжении всего ЖЦ, пользователи в большей степени будут довольны полученными результатами;
- ожидаемое качество продукта определяется при активном участии пользователя в процессе на ранних фазах разработки;
- благодаря меньшему объему доработок уменьшаются затраты на разработку;
- обеспечивается управление рисками;
- документация сконцентрирована на конечном продукте, а не на его разработке.

Недостатки макетирования

- требуется активное участие заказчика;
- на заказчиков может оказать негативное влияние тот факт, что они не располагают информацией о точном количестве итераций, которые будут необходимы;
- заказчик может предпочесть получить прототип, вместо того, чтобы ждать появления полной, хорошо продуманной версии;
- с учетом создания рабочего прототипа, качеству всего ПО или долгосрочной эксплуатационной надежности может быть уделено недостаточно внимания.
- прототипирование вызывает зависимость и может продолжаться слишком долго;
- при использовании модели решение трудных проблем может отодвигаться на будущее;
- при выборе инструментальных средств прототипирования (операционные системы, языки и малопродуктивные алгоритмы) разработчики могут остановить свой выбор на менее подходящем решении, только чтобы продемонстрировать свои способности.

Критерии применения макетирования

- требования не известны заранее, не постоянны или могут быть неверно истолкованы или неудачно сформулированы, требуют уточнения;
- выполняется новая, не имеющая аналогов разработка;
- если о прикладной программе отсутствует четкое представление;
- разработчики не уверены в том, какую оптимальную архитектуру или алгоритмы следует применять;
- существует потребность в разработке пользовательских интерфейсов;
- нужна проверка концепции;
- осуществляются временные демонстрации;
- если часть информационной системы требует прототипирования;
- требуется продемонстрировать техническую осуществимость, когда технический риск высок.

Инкрементная модель жизненного цикла

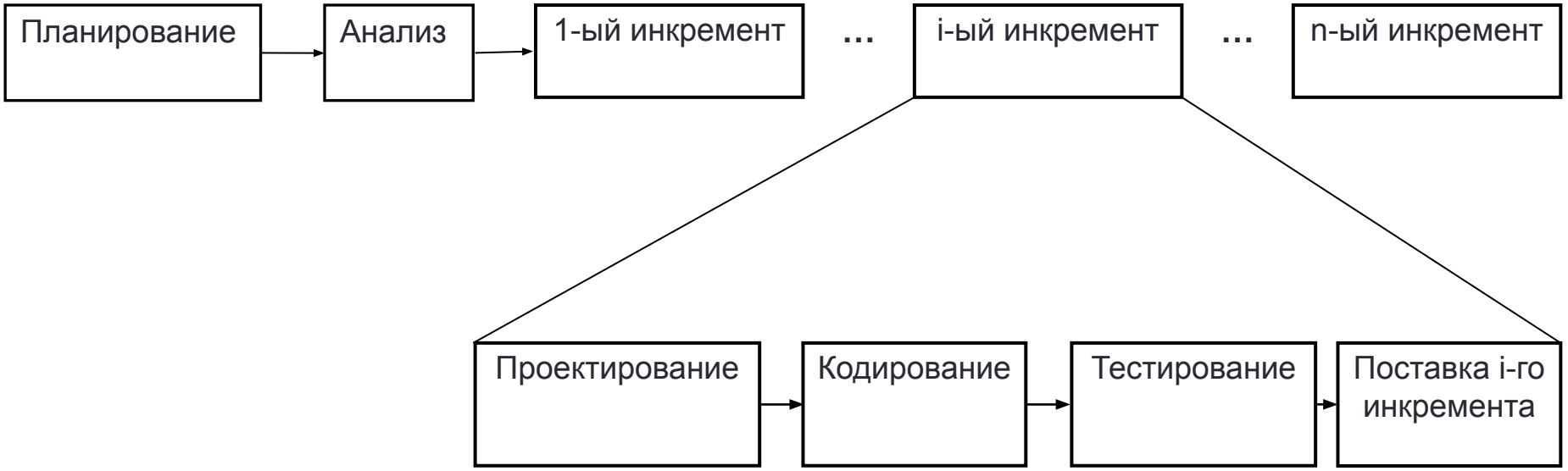
Инкрементная разработка представляет собой процесс частичной реализации всей системы и медленного наращивания функциональных возможностей.

Инкрементная модель действует по принципу каскадной модели с перекрытиями.

Два подхода к набору требований:

- полный заранее сформированный набор требований, которые выполняются в виде последовательных, небольших по размеру проектов,
- выполнение проекта может начаться с формулирования общих целей, которые затем уточняются и реализуются группами разработчиков.

Инкрементная модель ЖЦ



Преимущества инкрементной модели

- в результате выполнения каждого инкремента получается функциональный продукт;
- заказчик располагает возможностью высказаться по поводу каждой разработанной версии системы;
- правило по принципу "разделяй и властвуй" позволяет разбить возникшую проблему на управляемые части;
- заказчики могут распознавать самые важные и полезные функциональные возможности продукта на более ранних этапах разработки;
- требования стабилизируются на момент создания определенного инкремента;
- инкременты функциональных возможностей несут больше пользы и проще при тестировании
- снижается риск неудачи и изменения требований;
- риск распределяется на несколько меньших по размеру инкрементов;
- существует возможность пересмотреть риски, связанные с затратами и соблюдением установленного графика;
- заказчик может привыкать к новой технологии постепенно.

Недостатки инкрементной модели

- определение полной функциональной системы должно осуществляться в начале ЖЦ, чтобы обеспечить определение инкрементов;
- для модели необходимы хорошее планирование и проектирование;
- использование на этапе анализа общих целей, вместо полностью сформулированных требований, может оказаться неудобным для руководства;
- поскольку создание некоторых модулей будет завершено значительно раньше других, возникает необходимость в четко определенных интерфейсах;
- заказчик должен осознавать, что общие затраты на выполнение проекта не будут снижены;
- в модели не предусмотрены итерации в рамках каждого инкремента;
- формальный критический анализ и проверку намного труднее выполнить для инкрементов, чем для системы в целом;
- может возникнуть тенденция к оттягиванию решений трудных проблем на будущее с целью продемонстрировать руководству успех, достигнутый на ранних этапах разработки.

Критерии применения инкрементной модели

- если большинство требований можно сформулировать заранее, но их появление ожидается через определенный период времени;
- если рыночное окно слишком "узкое" и существует потребность быстро поставить на рынок продукт, имеющий функциональные базовые свойства;
- при разработке программ, связанных с низкой или средней степенью риска;
- при выполнении проекта с применением новой технологии, что позволяет пользователю адаптироваться к системе путем выполнения более мелких инкрементных шагов, без резкого перехода к применению основного нового продукта;
- когда однопроходная разработка системы связана с большой степенью риска.

Различие инкрементной и эволюционной моделей

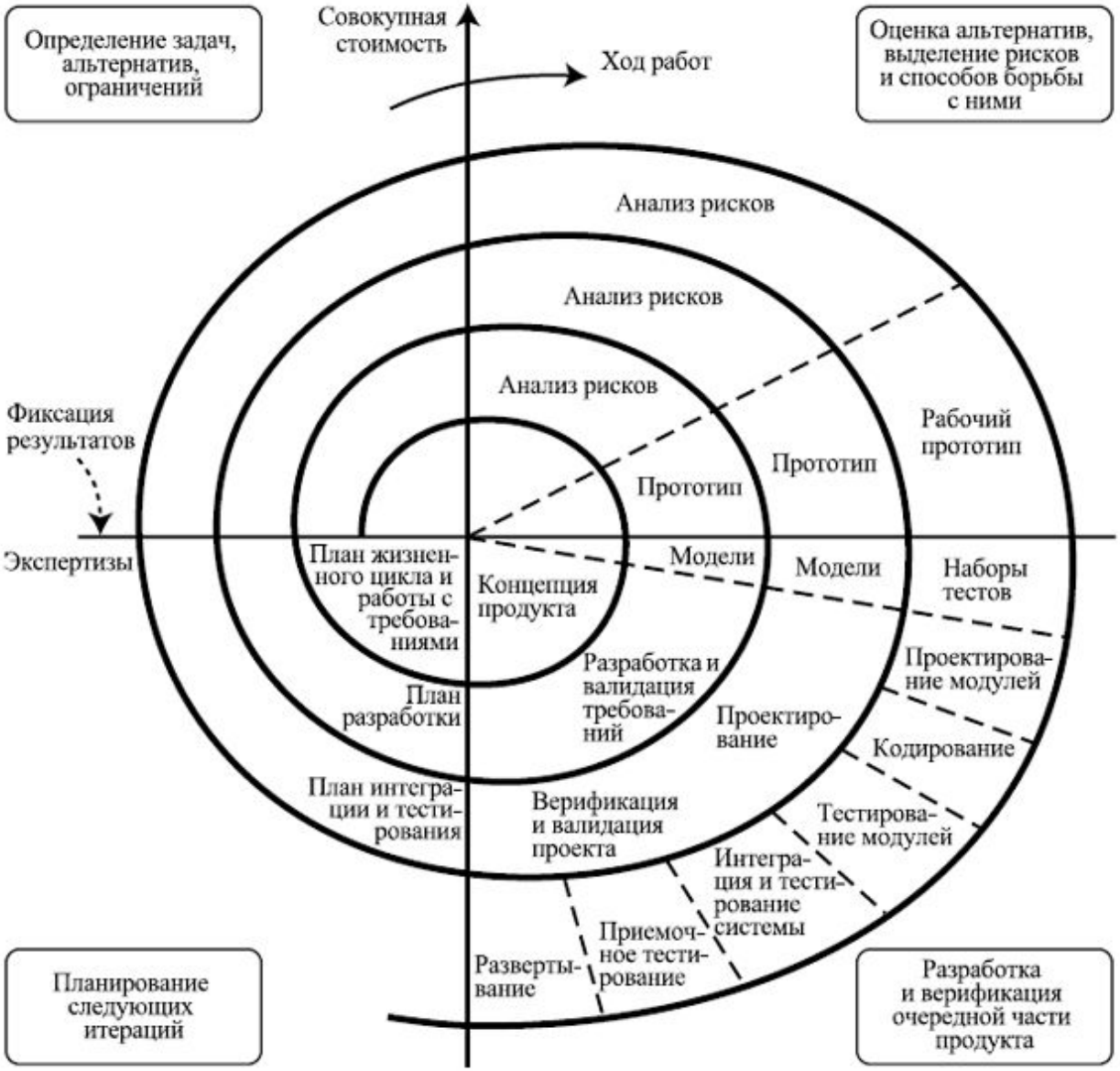
- Инкрементная модель
- Эволюционная модель

Спиральная модель

Спиральная модель (автор: Барри Боэм, 1988) является реализацией эволюционной стратегии разработки программного обеспечения.

Спиральная модель отображает базовую концепцию, которая заключается в том, что каждый цикл представляет собой набор операций, которому соответствует такое же количество стадий, как и в модели каскадного процесса.

Спиральная модель



Спиральная модель

Определение задач, альтернатив, ограничений:

- Выполняется определение целей (рабочая характеристика, выполняемые функции, возможность внесения изменений, решающих факторов достижения успеха и аппаратного/программного интерфейса).
- Определяются альтернативные способы реализации этой части продукта (конструирование, повторное использование, покупка, субдоговор, и т.п.). Определяются ограничения, налагаемые на применение альтернативных вариантов (затраты, график выполнения, интерфейс, ограничения, относящиеся к среде и др.).
- Создается документация, подтверждающая риски, связанные с недостатком опыта в данной сфере, применением новой технологии, жесткими графиками, плохо организованными процессами и т.д.;

Спиральная модель

Оценка альтернатив, выделение рисков и способов борьбы с ними:

- Выполняется оценка альтернативных вариантов, относящихся к целям и ограничениям.
- Выполняется определение и разрешение рисков.
- Принятие решения о прекращении/продолжении работ над проектом.

Разработка и верификация очередной версии продукта:

- создание проекта;
- критический анализ проекта;
- разработка кода;
- проверка кода;
- тестирование и компоновка продукта.

Спиральная модель

Планирование следующих итераций:

- разработка плана проекта,
- разработка плана менеджмента конфигурацией,
- разработка плана тестирования;
- разработка плана установки программного продукта.

Преимущества спиральной модели

- позволяет пользователям "увидеть" систему на ранних этапах;
- она обеспечивает разбиение большого потенциального объема работы по разработке продукта на небольшие части;
- в модели предусмотрена возможность гибкого проектирования;
- реализованы преимущества инкрементной модели (выпуск инкрементов, сокращение графика посредством перекрывания инкрементов);
- быстрая обратная связь по направлению от пользователей к разработчикам;
- обеспечивается определение непреодолимых рисков без особых дополнительных затрат;
- эта модель разрешает пользователям активно принимать участие при планировании, анализе рисков, разработке, а также при выполнении оценочных действий;
- при использовании спиральной модели не нужно распределять заранее все необходимые для выполнения проекта финансовые ресурсы.

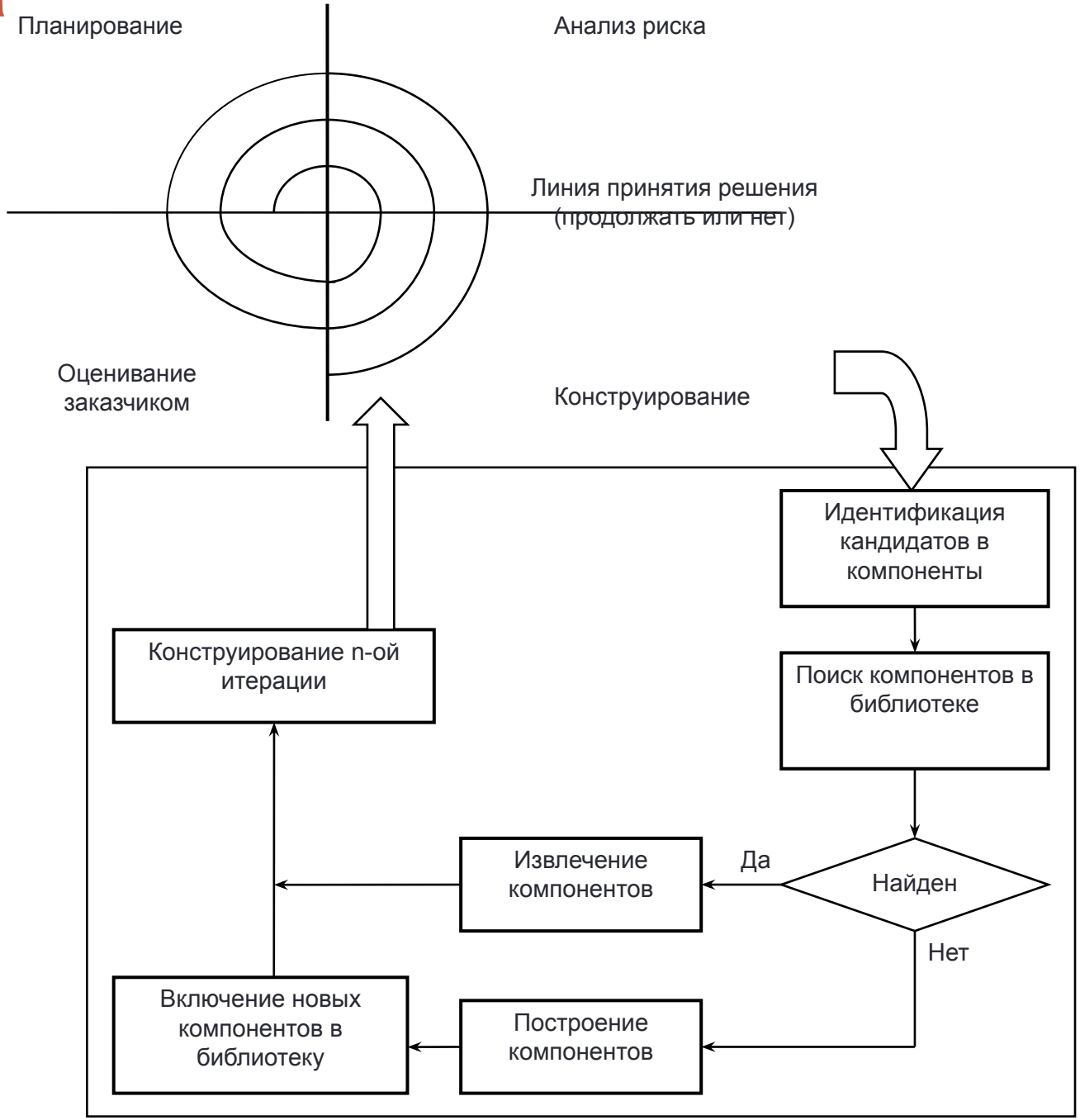
Недостатки спиральной модели

- модель имеет усложненную структуру, поэтому может быть затруднено ее применение разработчиками, менеджерами и заказчиками;
- спираль может продолжаться до бесконечности, поскольку каждая ответная реакция заказчика на созданную версию может порождать новый цикл;
- могут возникнуть затруднения при определении целей и стадий, указывающих на готовность продолжать процесс разработки на следующей итерации;
- большое количество промежуточных стадий может привести к необходимости в обработке внутренней дополнительной и внешней документации;
- если проект имеет низкую степень риска или небольшие размеры, модель может оказаться дорогостоящей;
- отсутствие хорошего средства или метода прототипирования может сделать использование модели неудобным.

Критерии применения спиральной модели

- когда создание прототипа представляет собой подходящий тип разработки продукта;
- когда пользователи не уверены в своих потребностях или требования слишком сложные;
- когда речь идет о применении новой технологии и когда необходимо протестировать базовые концепции;
- при разработке новой функции или новой серии продуктов;
- для проектов, выполнение которых сопряжено со средней и высокой степенью риска;
- для организаций, которые не могут себе позволить выделить заранее все необходимые для выполнения проекта денежные средства, и когда в процессе разработки отсутствует финансовая поддержка;
- когда нет смыла браться за выполнение долгосрочного проекта из-за потенциальных изменений, которые могут произойти в экономических приоритетах;
- с целью демонстрации качества и достижения целей за короткий период времени.

Компонентная модель



Модель быстрой разработки приложений

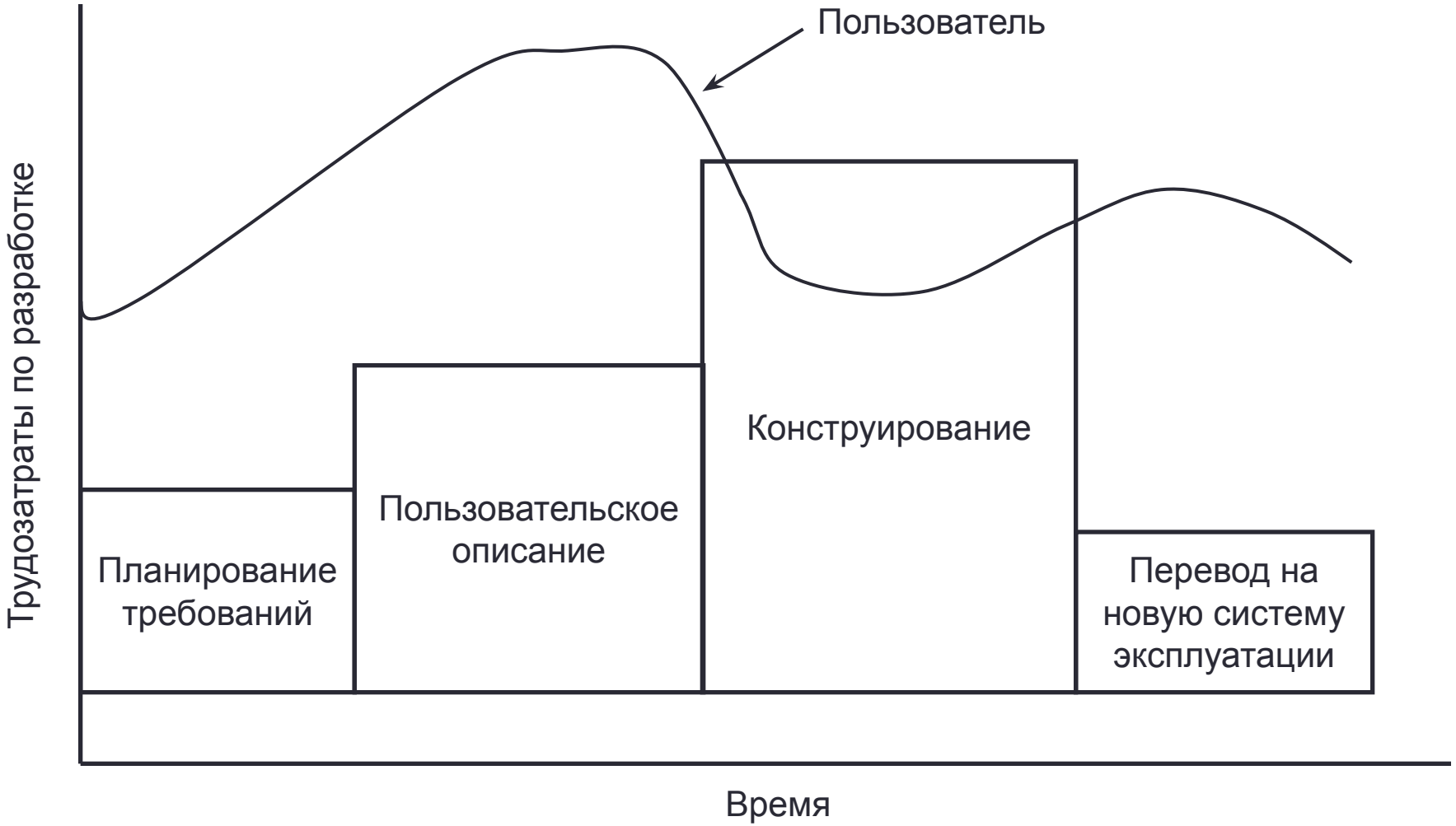
- Благодаря методу RAD (Rapid Application Developmment) пользователь задействован на всех фазах ЖЦ разработки проекта – не только при определении требований, но и при проектировании, разработке, тестировании, а также конечной поставке программного продукта.
- Характерной чертой RAD является короткое время перехода от определения требований до создания полной системы. Метод основывается на последовательности итераций эволюционной системы или прототипов, критический анализ которых обсуждается с заказчиком. В процессе такого анализа формируются требования к продукту.
- Разработка каждого интегрированного продукта ограничивается четко определенным периодом времени, который, как правило, составляет 60 дней и называется временным блоком.

Модель быстрой разработки приложений

Фазы RAD:

- **этап планирования требований** — сбор требований выполняется при использовании рабочего метода, называемого совместным планированием требований (Joint requirements planning, JRP), который представляет собой структурный анализ и обсуждение имеющихся коммерческих задач;
- **пользовательское описание** — совместное проектирование приложения (Joint application design, JAD) используется с целью привлечения пользователей;
- **фаза конструирования —** эта фаза объединяет в себе детализированное проектирование, построение (кодирование и тестирование), а также поставку программного продукта заказчику за определенное время;
- **перевод на новую систему эксплуатации** — эта фаза включает проведение пользователями приемочных испытаний, установку системы и обучение пользователей.

Модель быстрой разработки



Преимущества RAD

- время цикла разработки сокращается благодаря использованию мощных инструментальных средств;
- требуется меньшее количество специалистов;
- существует возможность произвести быстрый изначальный просмотр продукта;
- уменьшаются затраты;
- благодаря принципу временного блока уменьшаются затраты и риск, связанный с соблюдением графика;
- обеспечивается эффективное использование имеющихся в наличии средств и структур;
- постоянное присутствие заказчика сводит до минимума риск неудовлетворения продуктом и гарантирует соответствие системы коммерческим потребностям и надёжности программного продукта в эксплуатации;

Преимущества RAD

- основное внимание переносится с документации на код, причем при этом справедлив принцип "получаете то, что видите" (What you see is what you get, WYSIWYG);
- в модели используются следующие принципы и инструментальные средства моделирования:
  - ✔ деловое моделирование (методы передачи информации, место генерирования информационных потоков, кем и куда направляется, каким образом обрабатывается);
  - ✔ моделирование данных (происходит идентификация объектов данных и атрибутов, а также взаимосвязей);
  - ✔ моделирование процесса (выполняется преобразование объектов данных);
  - ✔ генерирование приложения (методы четвертого поколения);
- повторное использование компонент уже существующих программ.

Недостатки RAD

- непостоянное участие пользователя может негативно сказаться на конечном продукте;
- для реализации модели требуются разработчики и заказчики, которые готовы к быстрому выполнению действий ввиду жестких временных ограничений;
- при использовании этой модели необходимо достаточное количество высококвалифицированных разработчиков;
- использование модели может оказаться неудачным в случае отсутствия пригодных для повторного использования компонент;
- при использовании модели "вслепую" на затраты и дату завершения работы над проектом ограничения не накладываются;
- искусственное «затягивание» разработки ПО;
- существует риск, что работа над проектом никогда не будет завершена.

Критерии применения RAD

- в системах, которые поддаются моделированию, а также в масштабируемых системах;
- в системах, требования для которых в достаточной мере хорошо известны;
- в информационных системах;
- в случаях, когда конечный пользователь может и хочет принимать участие в процессе разработки на протяжении всего ЖЦ;
- при невысокой степени технических рисков;
- при выполнении проектов, разработка которых должна быть выполнена в сокращенные сроки (как правило, не более, чем за 60 дней);
- в системах, которые предназначены для концептуальной проверки, являются некритическими или имеют небольшой размер;
- когда затраты и соблюдение графика не являются самым важным вопросом (например при разработке внутренних инструментальных средств).

Выбор модели жизненного цикла

Процесс выбора:

1. Проанализируйте следующие отличительные категории проекта, помещенные в таблицах:
   - Требования.
   - Команда разработчиков.
   - Коллектив пользователей.
   - Тип проекта и риски.
2. Ответьте на вопросы, приведенные для каждой категории, обведя кружочком слова "да" или "нет".
3. Расположите по степени важности категории или вопросы, относящиеся к каждой категории, относительно проекта, для которого выбирается приемлемая модель.
4. Воспользуйтесь упорядоченными категориями для разрешения противоречий, возникающих при сравнении моделей, если общие полученные показатели сходны или одинаковы.

Таблица 1: Характеристика требований

Требования	Каскадная	V-образная	Прототи-пирование	Спираль-ная	RAD	Инкремент-ная
Являются ли требования легко определимыми и/или хорошо известными?	Да	Да	Нет	Нет	Да	Нет
Могут ли требования заранее определяться в цикле?	Да	Да	Нет	Нет	Да	Да
Часто ли будут изменяться требования в цикле?	Нет	Нет	Да	Да	Нет	Нет
Нужно ли демонстрировать требования с целью определения?	Нет	Нет	Да	Да	Да	Нет
Требуются ли для демонстрации возможностей проверка концепции?	Нет	Нет	Да	Да	Да	Нет
Будут ли требования отражать сложность системы?	Нет	Нет	Да	Да	Нет	Да
Обладает ли требование функциональными свойствами на раннем этапе?	Нет	Нет	Да	Да	Да	Да

Таблица 2: Характеристики команды разработчиков

Команда разработчиков проекта	Каскад-ная	V-образ-ная	Прототи-пирование	Спираль-ная	RAD	Инкре-ментная
Являются ли проблемы предметной области проекта новыми для большинства разработчиков?	Нет	Нет	Да	Да	Нет	Нет
Является ли технология предметной области проекта новой для большинства разработчиков?	Да	Да	Нет	Да	Нет	Да
Являются ли инструменты, используемые проектом, новыми для большинства разработчиков?	Да	Да	Нет	Да	Нет	Нет
Изменяются ли роли участников проекта во время жизненного цикла?	Нет	Нет	Да	Да	Нет	Да
Могут ли разработчики проекта пройти обучение?	Нет	Да	Нет	Нет	Да	Да
Является ли структура более значимой для разработчиков, чем гибкость?	Да	Да	Нет	Нет	Нет	Да
Будет ли менеджер проекта строго отслеживать прогресс команды?	Да	Да	Нет	Да	Нет	Да
Важна ли легкость распределение ресурсов?	Да	Да	Нет	Нет	Да	Да
Приемлет ли команда равноправные обзоры и инспекции, менеджмент/обзоры заказчика, а также стадии?	Да	Да	Да	Да	Нет	Да

Таблица 3: Характеристика коллектива пользователей

Коллектив пользователей	Каскад-ная	V-образ-ная	Прототи-пирование	Спираль-ная	RAD	Инкре-ментная
Будет ли присутствие пользователей ограничено в жизненном цикле?	Да	Да	Нет	Да	Нет	Да
Будут ли пользователи знакомы с определением системы?	Нет	Нет	Да	Да	Нет	Да
Буду ли пользователи ознакомлены с проблемами предметной области?	Нет	Нет	Да	Нет	Да	Да
Будут ли пользователи вовлечены во все фазы жизненного цикла?	Нет	Нет	Да	Нет	Да	Нет
Будет ли заказчик отслеживать ход выполнения проекта?	Нет	Нет	Да	Да	Нет	Нет

Таблица 4: Характеристика типов проектов и рисков

Тип проекта и риски	Каскадная	V-образная	Прототипирование	Спиральная	RAD	Инкрементная
Будет ли проект идентифицировать новое направление продукта для организации?	Нет	Нет	Да	Да	Нет	Да
Будет ли проект иметь тип системной интеграции?	Нет	Да	Да	Да	Да	Да
Будет ли проект являться расширением существующей системы?	Нет	Да	Нет	Нет	Да	Да
Будет ли финансирование проекта стабильным на всем протяжении жизненного цикла?	Да	Да	Да	Нет	Да	Нет
Ожидается ли длительная эксплуатация продукта в организации?	Да	Да	Нет	Да	Нет	Да
Должна ли быть высокая степень надежности?	Нет	Да	Нет	Да	Нет	Да
Будет ли система изменяться, возможно, с применением непредвиденных методов, на этапе сопровождения?	Нет	Нет	Да	Да	Нет	Да
Является ли график ограниченным?	Нет	Нет	Да	Да	Да	Да
Являются ли "прозрачными" интерфейсные модули?	Да	Да	Нет	Нет	Нет	Да
Доступны ли повторное используемые компоненты?	Нет	Нет	Да	Да	Да	Нет
Являются ли достаточными ресурсы (время, деньги, инструменты, персонал)?	Нет	Нет	Да	Да	Нет	Нет

Процесс выбора и подгонки модели ЖЦ

1. Ознакомьтесь с различными моделями.
2. Просмотрите и проанализируйте возможные виды работ: разработка, модернизация, сопровождение и т.д.
3. Выберите самый подходящий жизненный цикл, используя для этого матрицы критериев.
4. Проанализируйте, насколько выбранный жизненный цикл соответствует стандартам вашей организации, ваших заказчиков или типа проекта — ISO, IEEE и т.д.
5. Сформулируйте набор фаз и действий, образующих каждую фазу.
6. Определите внутренние и внешние производимые продукты.
7. Определите шаблоны и внутреннее содержимое поставляемых продуктов.
8. Определите действия по обзору, инспектированию, верификации и аттестации, а также стадии проекта.
9. Выполните оценку эффективности схемы жизненного цикла и проведите ее модернизацию там, где это необходимо.

Домашняя работа:

- Изучите теоретический материал.
- **Ответьте на контрольные вопросы (письменно):**
- **Блок 1:**

1. Опишите структуру жизненного цикла программного обеспечения (ЖЦПО). Назовите и опишите процессы ЖЦПО.
2. Назовите этапы ЖЦПО, опишите, что осуществляется на каждом этапе и что является результатом каждого этапа.
3. Назовите ГОСТы, необходимые для составления документации. Опишите их назначение.

- **Блок 2:**

1. Что понимается под моделью ЖЦПО, какие модели ЖЦ вы знаете?
2. Охарактеризуйте каскадную модель разработки ПО, приведите схему, укажите достоинства и недостатки.
3. Охарактеризуйте спиральную модель разработки ПО, приведите схему, укажите достоинства и недостатки.
4. Охарактеризуйте V-образную модель разработки ПО, приведите схему, укажите достоинства и недостатки.
5. Охарактеризуйте RAD модель разработки ПО, приведите схему, укажите достоинства и недостатки.
6. Охарактеризуйте инкрементную модель разработки ПО, приведите схему, укажите достоинства и недостатки.