Лекции 7_8

Модели жизненного цикла программного обеспечения. Каскадная модель и ее модификации. Модель прототипирования жизненного цикла разработки ПО. Модель быстрой разработки приложений RAD. Инкрементная и спиральная модели жизненного цикла разработки ПО

Разработка ПО

Стандарт ISO/IEC 12207 не предлагает конкретные методы разработки программного обеспечения: его положения являются общими для любых проектов по созданию ПО. Для каждого конкретного программного проекта необходимо разработать (а чаще - отобрать и адаптировать) свою модель жизненного цикла

Назначение моделей жизненного цикла ПО

- Модель жизненного цикла дает рекомендации по организации процесса разработки ПО в целом, конкретизируя его до видов деятельности, артефактов, ролей и их взаимосвязей
- Модель жизненного цикла служит основой для планирования программного проекта, позволяет экономнее расходовать ресурсы и добиваться более высокого качества управления
- Знание технологических функций, которые на разных этапах должны выполнять разработчики, занимающие те или иные роли, способствует правильному распределению обязанностей сотрудников

Знание моделей жизненного цикла помогает понять даже непрофессионалу, на что можно рассчитывать при заказе или приобретении программного обеспечения, а что нереально требовать от него

Организационные аспекты, подлежащие рассмотрению при формировании ЖЦ ПО

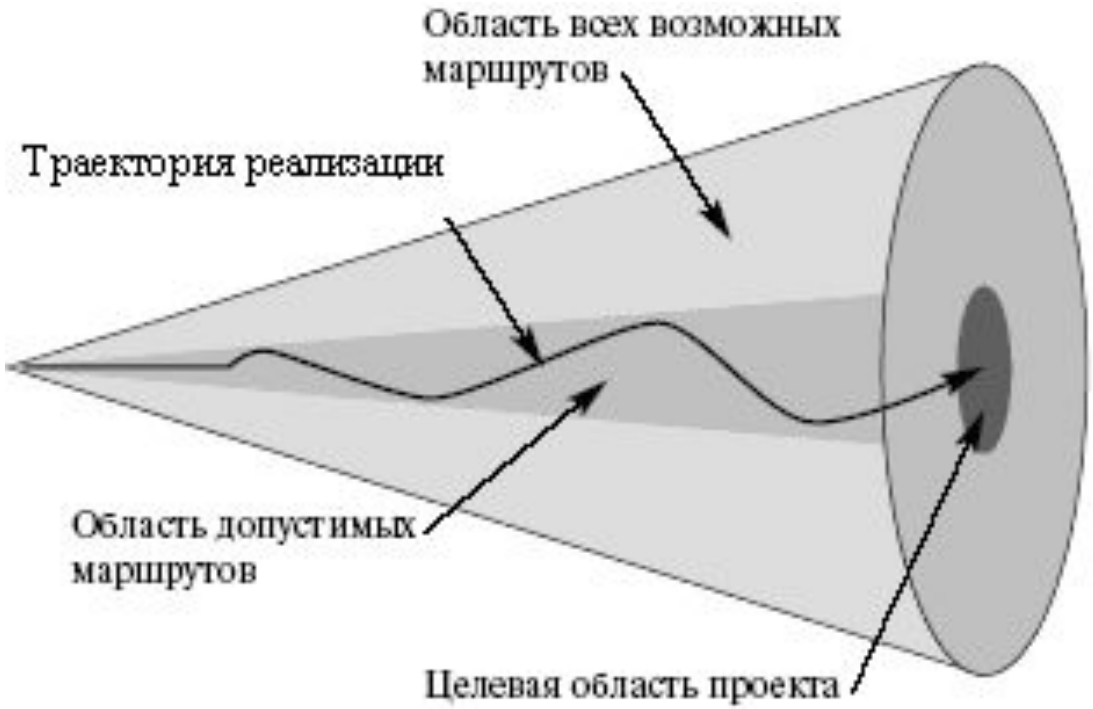
- планирование последовательности работ и сроков их исполнения
- подбор и подготовка ресурсов (людских, программных и технических) для выполнения работ
- оценка возможностей реализации проекта в заданные сроки, в пределах указанной стоимости и др.

Модель жизненного цикла ПО

Модель жизненного цикла программного обеспечения представляет собой **совокупность упорядоченных во времени, взаимосвязанных и объединенных в стадии работ, выполнение которых необходимо и достаточно для создания ПО, соответствующего заданным требованиям**

Модель жизненного цикла зависит от специфики, масштаба и сложности проекта, а также от условий, в которых система создается и функционирует. Каждая модель ЖЦ увязывается с конкретными методиками разработки программных систем и соответствующими стандартами в области программной инженерии

Конус операционных маршрутов проекта по разработке ПО

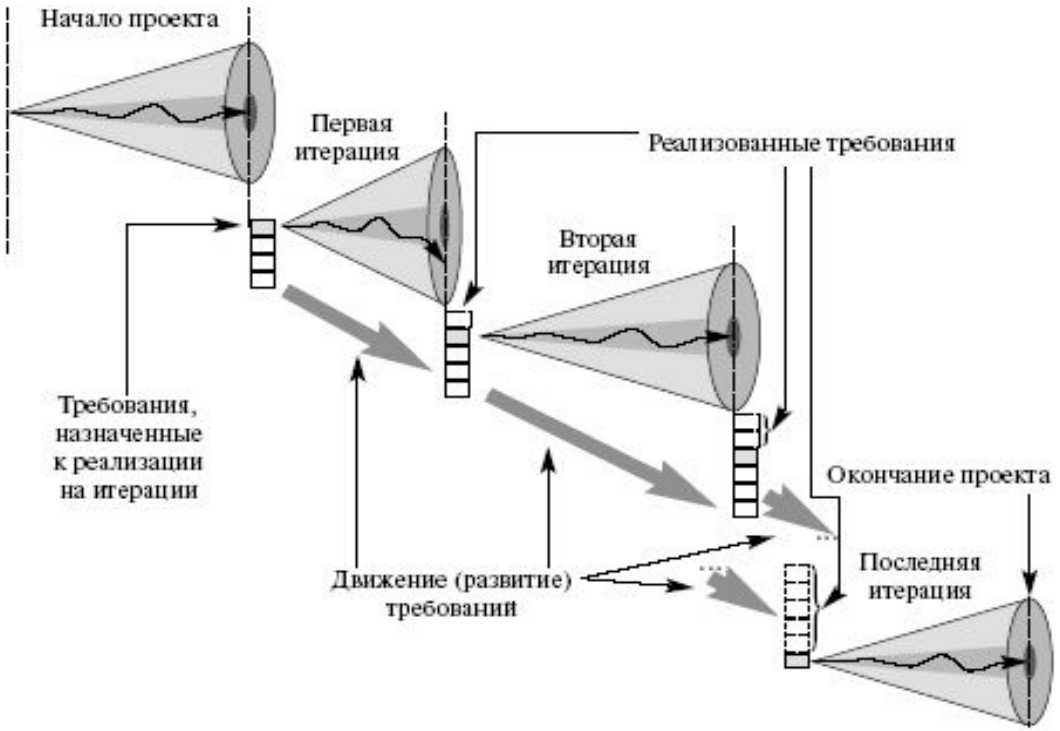


Последовательное развитие проекта

Каждый этап характеризуется:

- субъектом-исполнителем;
- сроками, когда должны быть решены задачи;
- выделенными ресурсами;
- средствами, инструментами и методами решения задач;
- контрольными мероприятиями, позволяющими удостовериться, что задачи этапа решены

Итеративное наращивание возможностей системы



Примечание:

пунктиром выделены отработанные требования

Рост количества прямоугольников при переходе от итерации к итерации отражает поступление новых требований в ходе развития проекта

Каскадная модель жизненного цикла

Переход на следующую стадию осуществляется только после полного завершения работ на текущей стадии, возвратов на пройденные стадии не предусматривается

Строго последовательное и однократное выполнение всех фаз проекта с детальным предварительным планированием в условиях определенных и зафиксированных требований к программной системе

Достоинства каскадной модели

- модель хорошо известна потребителям, не имеющим отношения к разработке и эксплуатации программ, и конечным пользователям (она часто используется в организациях для отслеживания проектов, не связанных с разработкой ПО)
- она доступна для понимания, а также проста и удобна в применении, так как процесс разработки выполняется поэтапно
- структурой модели может руководствоваться даже слабо подготовленный в техническом плане или неопытный персонал
- она отличается стабильностью требований
- модель хорошо срабатывает тогда, когда требования к качеству доминируют над требованиями к затратам и графику выполнения проекта

Достоинства каскадной модели

- каскадная модель способствует осуществлению строгого контроля менеджмента проекта, облегчая работу по составлению плана и комплектации команды разработчиков
- она позволяет участникам проекта, завершившим действия на выполняемой ими фазе, принять участие в реализации других проектов
- ход выполнения проекта легко проследить с помощью использования временной шкалы (или диаграммы Ганта), поскольку момент завершения каждой фазы используется в качестве вехи, т.е. контрольной точки, позволяющей проанализировать достигнутые результаты

Недостатки каскадной модели

- при разработке ПО высок риск создания системы, не удовлетворяющей изменившимся потребностям пользователей:
  - пользователи не в состоянии сразу изложить все свои требования и не могут предвидеть как они изменятся в ходе разработки
  - за время разработки могут произойти изменения во внешней среде, которые повлияют на требования к системе
- согласование получаемых результатов с пользователями производится только в точках, планируемых после завершения каждой стадии

Недостатки каскадной модели

- спецификации программного обеспечения невозможно зафиксировать
- программные проекты не определяются трудозатратами
- основные риски программных проектов смещены на завершающий этап работ
- при разработке ПО порядок работ не является фиксированным, а программные продукты являются абстрактными

Где применяется каскадный процесс?

- в критически важных системах реального времени (таких как, например, управление авиационным движением или медицинским оборудованием)
- в масштабных проектах, в реализации которых задействовано несколько больших команд разработчиков
- при разработке новой версии уже существующего продукта или переносе его на новую платформу
- в организациях, имеющих большой практический опыт в создании программных систем определенного типа (например, бухгалтерский учет, начисление зарплаты и пр.)

V-образная модель жизненного цикла разработки ПО

Примечание: в модели выделены связи между шагами, предшествующими программированию, и соответствующими видами тестирования и испытаний. Пунктирные линии показывают, что соответствующие фазы выполняются параллельно

Фазы V-образной модели ЖЦ

- *Планирование проекта и требований*: определяются системные требования и устанавливаются правила распределения ресурсов. В случае необходимости выделяются функции для аппаратного и программного обеспечения
- *Анализ требований к продукту*: разрабатывается полная спецификация программной системы
- *Разработка архитектуры или верхнеуровневое проектирование:* определяется состав компонентов и их взаимосвязи, а также распределение функций по компонентам системы
- *Детализированная разработка проекта:* определяются и документируются алгоритмы для каждого компонента, выделенного на фазе построения архитектуры
- *Кодирование:* преобразование алгоритмов в программный код
- М*одульное тестирование:* проверка каждого закодированного модуля на наличие ошибок
- *Интеграция и тестирование:* сборка модулей и интеграционное тестирование
- *Системное и приемочное тестирование:* проверка функционирования программной системы в целом в аппаратной среде в соответствии со спецификацией требований к программному обеспечению
- *Производства и эксплуатация:* программное обеспечение запускается в производство. На этой же фазе предусмотрены модернизация и внесение поправок в ПО

Достоинства V-образной модели

- модель проста в использовании (относительно проекта, для которого она является приемлемой)
- в модели предусмотрены аттестация и верификация всех внешних и внутренних полученных данных, а не только самого программного продукта
- модель определяет продукты, которые должны быть получены в результате процесса разработки, причем каждые полученные данные должны подвергаться тестированию
- благодаря модели легко отслеживать ход процесса разработки, так как в данном случае вполне возможно воспользоваться временной шкалой, а завершение каждой фазы рассматривать в качестве контрольной точки

Недостатки V-образной модели

- в модели не учтены итерации между фазами
- в модели не предусмотрено внесение 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
- при использовании модели часто бывает сложно справиться с параллельными событиями

Область применения V-образной модели

- когда вся информация о требованиях доступна заранее
- при разработке систем высокой надежности

V-образная модель применяется при создании:
• прикладных программ для наблюдения за пациентами в клиниках;
• встроенного ПО для устройств управления аварийными подушками безопасности в автомобилях

Упрощенный процесс системного проектирования

- сбор и анализ требований
- разработка архитектуры (определение и составление спецификаций систем)
- разработка подсистем
- интеграция и проверка

Недостатки процесса системного проектирования

- классическое системное проектирование определяется документацией
- интеграция является конечной, а не пошаговой операцией
- сложные и противоречивые взаимосвязи между генеральным подрядчиком и субподрядчиками

«Основное заблуждение каскадной модели состоит в предположениях, что проект проходит через весь процесс один раз, архитектура хороша и проста в использовании, проект осуществления разумен, а ошибки в реализации устраняются по мере тестирования. Иными словами, каскадная модель исходит из того, что все ошибки будут сосредоточены в реализации, а потому их устранение происходит равномерно во время тестирования компонентов и системы в целом»

Ф. Брукс

Причины использования эволюционных и инкрементных моделей жизненного цикла ПО

«В большинстве проектов первая построенная система едва ли пригодна к употреблению. Она может быть слишком медленной, слишком объемной, неудобной в использовании или обладать всеми тремя перечисленными недостатками. Нет другого выбора, кроме как начать с самого начала, приложив все усилия, и построить модернизированную версию, в которой решались бы все три проблемы.

В случае, когда в проекте используется новая системная концепция или новая технология, разработчик вынужден построить систему, которой впоследствии не воспользуется, поскольку даже при наилучшем планировании невозможно предвидеть достижение нужного результата.

Следовательно, вопрос менеджмента заключается не в том, создавать или нет экспериментальную систему, которой затем не воспользуются. Вы в любом случае так и сделаете. Единственный вопрос в том, нужно ли планировать создание продукта одноразового использования заранее или обещать поставить его заказчикам...»

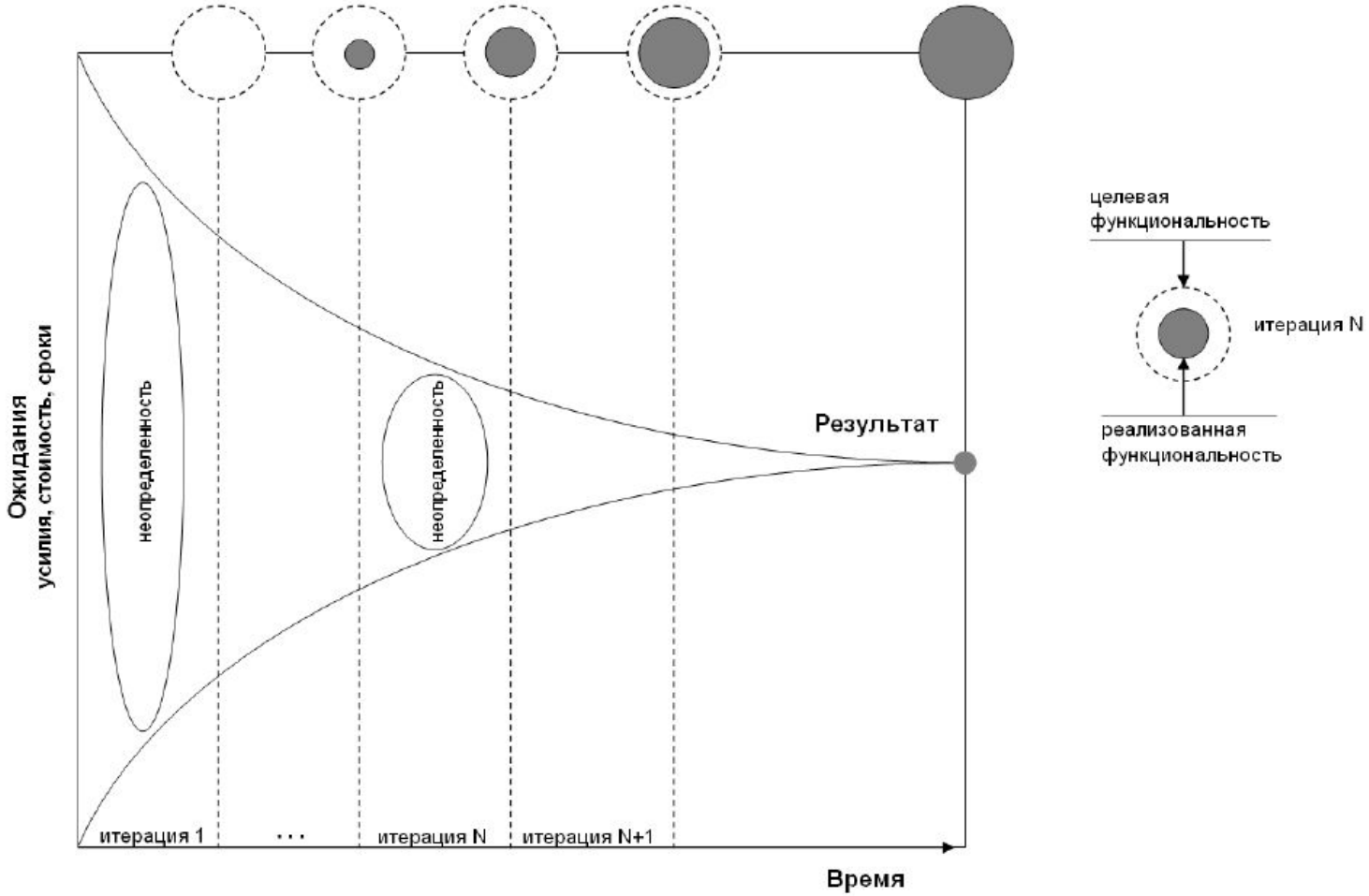
Ф. Брукс

Основные задачи, решаемые за счет создания прототипов:

- итеративное извлечение и уточнение требований к продукту
- возможность изменить эксплуатационное окружение программной системы
- снижение неопределенности (а следовательно, и рисков) по мере завершения каждой итерации

Прототип - легко поддающаяся модификации и расширению рабочая модель разрабатываемого программного средства (или системы), позволяющая пользователю получить представление о его ключевых свойствах до полной реализации

Снижение неопределенности и инкрементное расширение функциональности при итеративной организации жизненного цикла



Модель прототипирования жизненного цикла разработки ПО

Краткое описание процесса прототипирования

- Разработка предварительного плана проекта на основе предварительных требований
- Быстрый анализ с целью создания высокоуровневой модели системы, которая используется для построения исходного прототипа в объеме проектирования базы данных и пользовательского интерфейса, а также разработки некоторых функций
- Итерационный процесс быстрого прототипирования, в ходе которого разработчики демонстрируют очередной прототип, а пользователи оценивают его функционирование. Обнаружившиеся проблемы устраняются совместными усилиями обеих сторон

Достоинства модели быстрого прототипирования

- взаимодействие пользователя и/или заказчика с системой начинается на раннем этапе разработки, что минимизирует возможность возникновения разногласий
- исходя из реакции заказчиков на демонстрацию текущего прототипа продукта, разработчики получают сведения о различных аспектах поведения системы, благодаря чему сводится к минимуму количество неточностей в требованиях и обеспечивается возможность внести новые требования
- возможность наблюдать ту или иную функцию в действии побуждает к разработке дополнительных функциональных возможностей
- модель позволяет выполнять гибкое проектирование и разработку, включая несколько итераций на всех фазах жизненного цикла
- благодаря меньшему объему доработок уменьшаются совокупные затраты на разработку и обеспечивается управление рисками
- документация сконцентрирована на конечном продукте, а не на процессе его разработки
- при использовании модели заказчикам демонстрируются постоянные, видимые признаки прогресса в реализации проекта, что дает им возможность чувствовать себя более уверенно
- принимая участие в процессе разработки на протяжении всего жизненного цикла, пользователи в большей степени будут удовлетворены полученными результатами

Недостатки модели быстрого прототипирования

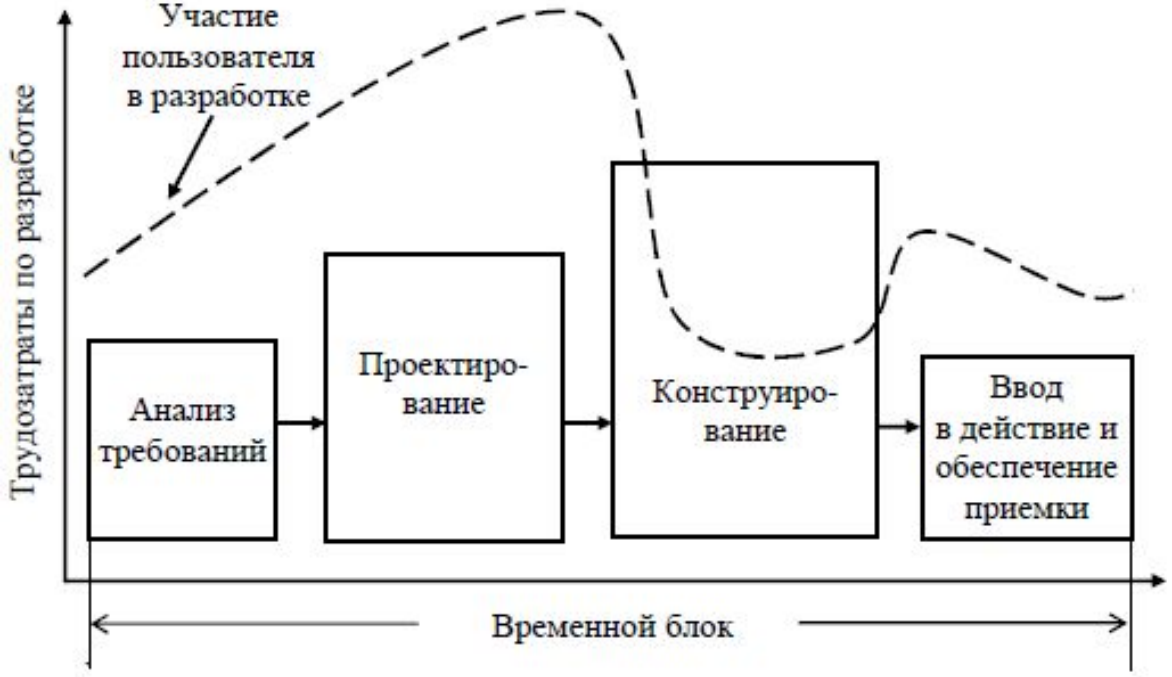
- модель может быть отклонена из-за сложившейся репутации о ней как о методе разработки «на скорую руку»
- с учетом создания рабочего прототипа, качеству всего ПО или долгосрочной эксплуатационной надежности может быть уделено недостаточно внимания
- при использовании модели решение трудных проблем может отодвигаться на будущее. Это может привести к тому, что результирующие программные продукты не оправдают надежд, которые возлагались на прототип
- на итерационном этапе быстрый прототип представляет собой частичную реализацию системы, поэтому если выполнение проекта завершается досрочно, у конечного пользователя останется лишь незавершенный продукт
- разработанные прототипы часто страдают от неадекватной или недостающей документации
- заказчик может предпочесть получить прототип, вместо того, чтобы ждать появления полной, хорошо продуманной версии

Случаи применения модели быстрого прототипирования

- требования не известны заранее, или непостоянны, они могут быть неверно истолкованы или неудачно сформулированы
- существует потребность в разработке пользовательских интерфейсов
- нужна проверка концепции
- выполняется новая, не имеющая аналогов разработка (в отличие от эксплуатации продукта на уже существующей системе)
- разработчики не уверены в том, какую оптимальную архитектуру или алгоритмы следует применять
- алгоритмы или системные интерфейсы усложнены
- требуется продемонстрировать техническую осуществимость, когда технический риск высок
- в проектах по разработке ПО со средней и/или высокой степенью риска
- заказчик настаивает на временных демонстрациях прогресса разработки

Чаще всего модель прототипирования применяется в системах с ярко выраженным пользовательским интерфейсом

Модель быстрой разработки приложений RAD (Rapid Application Development)



В RAD-модели для ускорения процесса разработки обычно используются различные виды моделирований предметной области, например, функциональное моделирование на базе стандарта IDEF0, с помощью которого определяются и анализируются функции и информационные потоки предметной области

Базовые предпосылки метода:

- Создание последовательности прототипов, критический анализ которых производится пользователем (заказчиком)
- Пользователь (заказчик) задействован на всех фазах жизненного цикла разработки проекта
- Разработка продукта осуществляется в рамках временного блока, продолжительность которого составляет не более 60 дней

Фазы модели 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. На этой фазе проектирования системы, не являющейся промышленной, работающая над проектом команда зачастую использует автоматические инструментальные средства, обеспечивающие сбор пользовательской информации
- **фаза конструирования («до полного завершения») —** эта фаза объединяет в себе детальное проектирование, кодирование и тестирование, а также поставку программного продукта заказчику за определенное время. Сроки выполнения этой фазы в значительной мере зависят от использования генераторов кода, экранных генераторов и других типов производственных инструментальных средств
- **перевод на новую систему эксплуатации —** эта фаза включает проведение пользователями приемочных испытаний, установку системы и обучение пользователей

Достоинства модели RAD

- время цикла разработки сокращается благодаря использованию мощных инструментальных средств
- требуется меньшее количество специалистов (поскольку разработка системы выполняется усилиями команды, осведомленной в предметной области)
- благодаря принципу временного блока уменьшаются затраты и риск, связанный с соблюдением графика
- обеспечивается эффективное использование имеющихся в наличии ресурсов
- постоянное присутствие заказчика сводит до минимума риск неудовлетворенности продуктом и гарантирует соответствие системы коммерческим потребностям
- основное внимание переносится с документации на код, причем при этом справедлив принцип «получаете то, что видите» (What you see is what you get, WYSIWYG)
- допускается повторное использование ранее разработанных компонентов

Недостатки модели RAD

- для реализации модели требуются разработчики и заказчики, которые готовы к быстрому выполнению действий ввиду жестких временных ограничений
- невозможность участия пользователей на протяжении всего жизненного цикла может негативно сказаться на конечном продукте
- при использовании данной модели необходимо достаточное количество высококвалифицированных разработчиков, умеющих применять выбранные инструментальные средства для сокращения времени разработки
- использование модели может оказаться неудачным в случае отсутствия пригодных для повторного использования компонентов
- при отсутствии тесного взаимодействия менеджмента проекта как с командой разработчиков, так и с заказчиком, возможно возникновение замкнутого цикла разработки

Область применения модели RAD

- в системах, которые поддаются моделированию (основанных на использовании компонентных объектов), а также в масштабируемых системах
- в системах, требования для которых в достаточной мере известны
- в случаях, когда конечный пользователь может принимать участие в процессе разработки на протяжении всего жизненного цикла
- 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 случаях, когда пригодные к повторному использованию части можно получить из автоматических хранилищ программных продуктов
- 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)

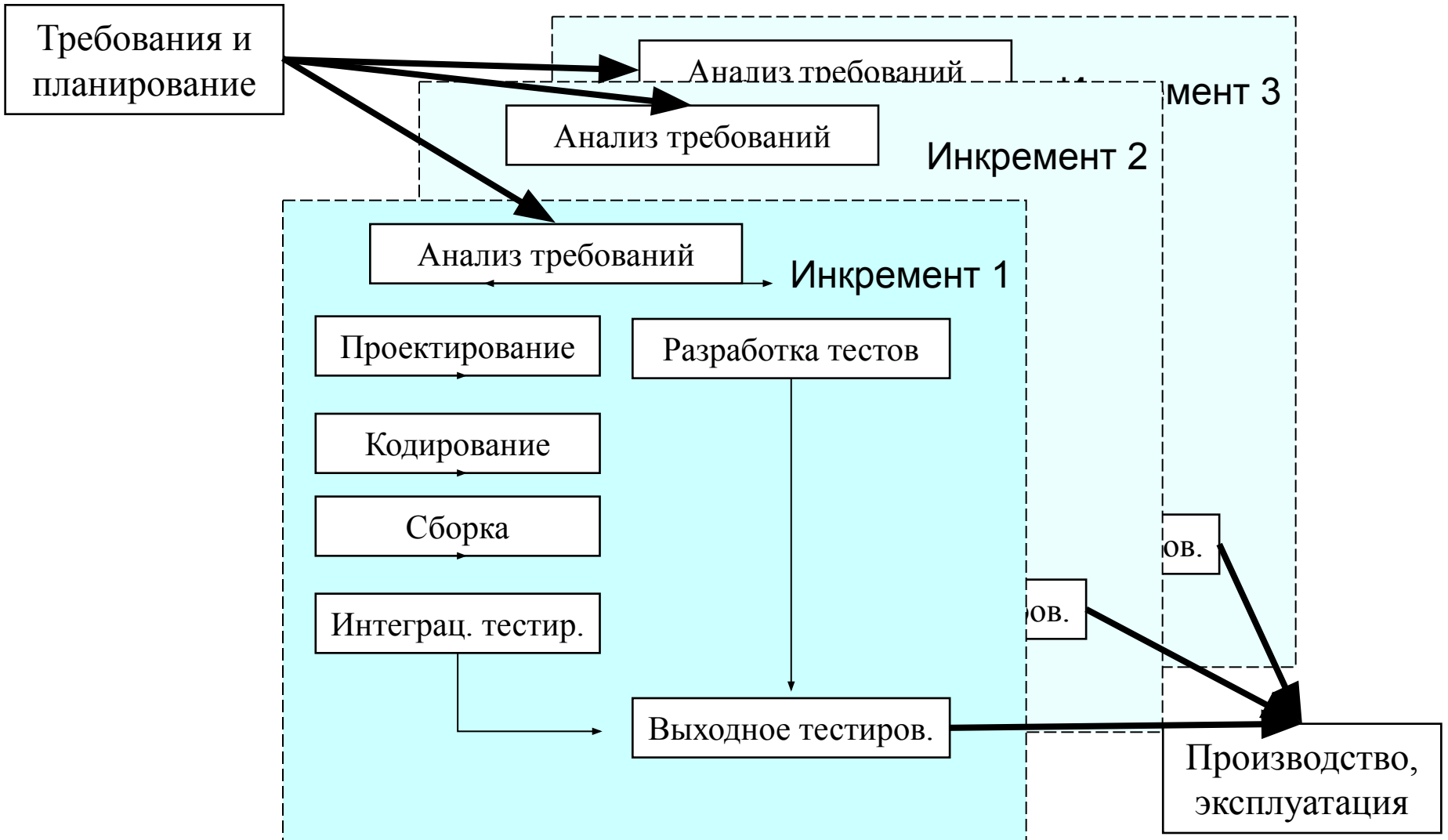
Модель RAD часто используется при создании информационных систем

Особенности инкрементной модели

- Инкрементная модель основана на запланированном улучшении продукта в процессе его ЖЦ
- При использовании инкрементных моделей осуществляется изначальная частичная реализация всей системы (или программного средства)
- В первую очередь реализуются базовые требования, затем следует медленное наращивание функциональных возможностей или характеристик качества ПО в реализуемых последовательно прототипах (инкрементах)

Особенностью инкрементной модели является большое количество циклов разработки при незначительной продолжительности цикла и небольших различиях между инкрементами соседних циклов

Инкрементная модель



Краткое описание инкрементной модели

- формулируется полный набор требований, которые по некоторому признаку делятся на части
- жизненный цикл проекта разбивается на последовательность итераций, каждая из которых представляет собой «мини-проект», включающий все фазы каскадной модели
- первый инкремент реализует базовые функции программного продукта, в последующих инкрементах функции программного средства постепенно расширяются, пока не будет реализован весь набор требований
- на каждой итерации получается работающая версия программной системы, обладающая функциональностью всех предыдущих плюс текущей итерации

Как правило, со временем инкременты уменьшаются и реализуют каждый раз меньшее количество требований

Достоинства инкрементной модели

- разработка ведется на основе стабильных требований, которые для каждого инкремента определяются заказчиком
- возможна оптимизация затрат, поскольку сначала выполняется разработка и реализация основной функции и/или функций из группы высокого риска
- снижаются затраты и ускоряется начальный график поставки продукта (что позволяет соответствовать возросшим требованиям рынка)
- в результате выполнения каждого инкремента получается работающий продукт, по поводу которого можно получить обратную связь от заказчика (пользователя)
- в конце каждой инкрементной поставки существует возможность пересмотреть риски, связанные с затратами и соблюдением установленного графика
- возможность начать построение следующей версии проекта на переходном этапе предыдущей версии сглаживает изменения, вызванные сменой персонала
- потребности клиента лучше поддаются управлению, поскольку время разработки каждого инкремента очень незначительно
- за счет последовательного внедрения инкрементов заказчик имеет возможность привыкать к новой технологии постепенно

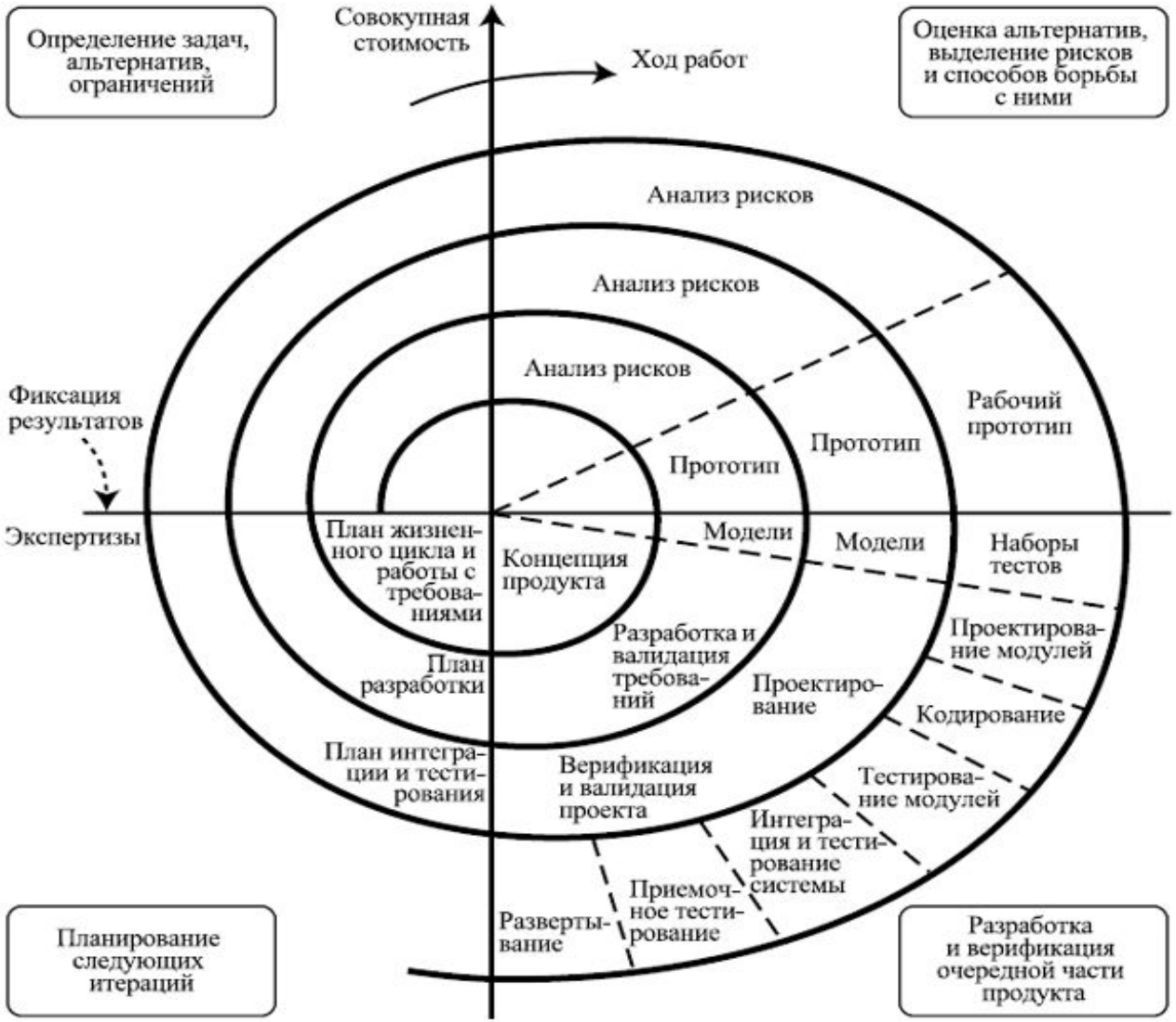
Недостатки инкрементной модели

- в модели не предусмотрены итерации в рамках каждого инкремента
- определение полной функциональности системы должно осуществляться в начале жизненного цикла, чтобы обеспечить определение инкрементов
- формальный критический анализ и проверку намного труднее выполнить для инкрементов, чем для системы в целом
- заказчик должен осознавать, что общие затраты на выполнение проекта не будут снижены
- поскольку создание некоторых модулей будет завершено значительно раньше других, возникает необходимость в четко определенных интерфейсах
- для модели необходимо хорошее планирование и проектирование
- может возникнуть тенденция к оттягиванию решений трудных проблем на будущее с целью продемонстрировать руководству успех, достигнутый на ранних этапах разработки

Область применения инкрементной модели

- большинство требований можно сформулировать заранее, но их появление ожидается через определенный период времени
- рыночное окно слишком «узкое» и существует потребность быстро поставить на рынок продукт, имеющий базовые функциональные свойства
- на выполнение проекта предусмотрен большой период времени, как правило, не меньше года
- степень важности различных свойств системы распределяется более менее равномерно
- 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
- когда результативные данные получаются через регулярные интервалы времени

Спиральная модель



Достоинства спиральной модели

- спиральная модель позволяет пользователям «увидеть» систему на ранних этапах разработки, что обеспечивается посредством использования ускоренного прототипирования в жизненном цикле разработки ПО
- модель обеспечивает возможность разработки сложной программной системы «по частям», выделяя на первых этапах наиболее значимые требования. Это позволяет в случае необходимости, прекратить работу над проектом, и уменьшить непроизводительные расходы
- в модели предусмотрена возможность гибкого проектирования, поскольку в ней воплощены преимущества каскадной модели, и в тоже время, разрешены итерации по всем фазам этой же модели
- модель позволяет идентифицировать риски без особых дополнительных затрат
- модель предусматривает активное участие пользователей в работах по планированию, анализу рисков, разработке и оценке полученных результатов. Обратная связь по направлению от пользователей к разработчикам выполняется с высокой частотой и на ранних этапах жизненного цикла, что обеспечивает создание нужного продукта высокого качества
- при использовании модели происходит усовершенствование процессов управления проектом, включая процесс обеспечения качества, соблюдение графика выполнения работ и использования ресурсов, что достигается путем выполнения обзора в конце каждой итерации
- модель обеспечивает повышение эффективности разработки благодаря использованию пригодных для повторного использования свойств
- при использовании спиральной модели повышается вероятность предсказуемого поведения системы за счет неоднократного уточнения поставленных целей
- модель позволяет выполнять частую оценку совокупных затрат, и, как следствие, контролировать степень риска

Недостатки спиральной модели

- модель имеет усложненную структуру, что иногда затрудняет ее применение разработчиками, менеджерами и заказчиками
- при использовании модели часто возникает сложность анализа и оценки рисков при выборе вариантов. Более того, если проект имеет низкую степень риска или небольшие размеры, модель может оказаться дорогостоящей, так как оценка рисков после прохождения каждого витка спирали связана с существенными затратами
- сложность поддержания версий продукта (хранение версий, возврат к ранним версиям, комбинация версий)
- сложность оценки точки перехода на следующий цикл
- спираль может продолжаться до бесконечности, поскольку каждая ответная реакция заказчика на созданную версию может порождать новый цикл, что отдаляет окончание работы над проектом

Спиральная модель жизненного цикла применяется в тех случаях, когда

- пользователи не уверены в своих потребностях или требования к системе слишком сложны и могут меняться в процессе выполнения проекта, что вызывает необходимость прототипирования для их анализа и оценки
- достижение успеха не гарантировано и необходима оценка рисков продолжения проекта
- проект является сложным, дорогостоящим и обосновать объемы финансирования возможно только в процессе его выполнения
- в проекте предусматривается применение новых технологий, что связано с риском их освоения и достижения ожидаемого результата
- при выполнении очень больших проектов, которые в силу ограниченности ресурсов можно выполнять только по частям

Область применения спиральной модели

- при разработке систем, требующих большого объема вычислений (например, систем принятия решений)
- при выполнении бизнес-проектов
- при выполнении проектов в области аэрокосмической промышленности, обороны и инжиниринга, где уже имеется позитивный опыт ее использования

Последовательная стратегия разработки ПО

- 1. Представляет собой однократный проход этапов разработки
- 2. Основана на полном определении всех требований к разрабатываемому программному средству или системе в начале процесса разработки
- 3. Каждый этап разработки начинается после завершения предыдущего этапа, возвратов к уже выполненным этапам не предусматривается
- 4. Промежуточные продукты разработки в качестве версии ПО (системы) не распространяются

Инкрементная стратегия разработки ПО

- 1. Многократный проход этапов разработки с запланированным улучшением результата
- 2. Основана на полном определении всех требований к разрабатываемому ПО (системе) в начале процесса разработки
- 3. Полный набор требований реализуется постепенно в соответствии с планом в последовательных циклах разработки. Результат каждого цикла называется инкрементом. Могут использоваться прототипы
- 4. Результат каждого цикла разработки может распространяться в качестве очередной поставляемой версии программного средства или системы

Эволюционная стратегия разработки ПО

- 1. Многократный проход этапов разработки
- 2. Частичное определение требований к разрабатываемому программному средству или системе в начале процесса разработки
- 3. Требования постепенно уточняются в последовательных циклах разработки, при этом часто используется прототипирование
- 4. Результат каждого цикла разработки представляет собой очередную поставляемую версию программного средства или системы
- 5. Включает существенно меньшее количество циклов разработки при большей продолжительности цикла по сравнению с икрементной стратегией

Модели жизненного цикла ПО

- Последовательные модели
  - Каскадная
  - V-образная
  - Упрощенный процесс системного проектирования
- Эволюционные и инкрементные модели
  - Модель прототипирования
  - Модель RAD
  - Инкрементная модель
  - Спиральная модель

Выбор модели жизненного цикла на основе характеристик требований

Требования	Каскадная	V-образная	Прототи-пирование	Спиральная	RAD	Инкре-ментная
Являются ли требования легко определимыми и/или хорошо известными?	Да	Да	Нет	Нет	Да	Нет
Могут ли требования быть определены заранее?	Да	Да	Нет	Нет	Да	Да
Часто ли будут изменяться требования на протяжении жизненного цикла?	Нет	Нет	Да	Да	Нет	Нет
Нужно ли демонстрировать требования с целью их определения?	Нет	Нет	Да	Да	Да	Нет
Требуется ли проверка концепции программного средства (системы)?	Нет	Нет	Да	Да	Да	Нет
Будут ли требования изменяться (уточняться) с ростом сложности системы (программного средства)?	Нет	Нет	Да	Да	Нет	Да
Нужно ли реализовать основные требования на ранних этапах разработки?	Нет	Нет	Да	Да	Да	Да

Выбор модели жизненного цикла на основе характеристик участников команды разработчиков

Команда разработчиков проекта	Каскадная	V-образная	Прототи-пирование	Спиральная	RAD	Инкре-ментная
Являются ли проблемы предметной области проекта новыми для большинства разработчиков?	Нет	Нет	Да	Да	Нет	Нет
Являются ли инструментальные средства, используемые в проекте, 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 процесса разработки более значимой для разработчиков, чем гибкость?	Да	Да	Нет	Нет	Нет	Да
Будет ли в проекте строго отслеживаться прогресс команды?	Да	Да	Нет	Да	Нет	Да
Важна ли легкость распределения человеческих ресурсов в проекте?	Да	Да	Нет	Нет	Да	Да
Приемлет ли команда разработчиков оценки, проверки, а также стадии работ?	Да	Да	Да	Да	Нет	Да

Выбор модели жизненного цикла на основе характеристик коллектива пользователей

Коллектив пользователей	Каскадная	V-образная	Прототипирование	Спиральная	RAD	Инкрементная
Будет ли присутствие пользователей ограничено в жизненном цикле?	Да	Да	Нет	Да	Нет	Да
Будут ли пользователи оценивать текущее состояние программного продукта (системы) в процессе раз работки?	Нет	Нет	Да	Да	Нет	Да
Будут 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

Выбор модели жизненного цикла на основе характеристик типа проектов и рисков

Тип проекта и риски	Каскадная	V-образная	Прототи-пирование	Спиральная	RAD	Инкре-ментная
Разрабатывается ли в проекте продукт нового для организации направления?	Нет	Нет	Да	Да	Нет	Да
Будет ли проект крупно- или среднемасштабным?	Нет	Да	Да	Да	Да	Да
Будет ли проект являться расширением существующей системы?	Да	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?	Да	Да	Нет	Да	Нет	Да
Должна ли быть высокая степень надежности?	Нет	Да	Нет	Да	Нет	Да
Велика ли вероятность изменения системы (продукта) на этапе сопровождения?	Нет	Нет	Да	Да	Нет	Да
Является ли график сжатым?	Нет	Нет	Да	Да	Да	Да
Являются ли "прозрачными" интерфейсные модули?	Да	Да	Нет	Нет	Нет	Да
Предполагается ли повторное использование компонентов?	Нет	Нет	Да	Да	Да	Нет
Являются ли достаточными ресурсы (время, деньги, инструменты, персонал)?	Нет	Нет	Да	Да	Нет	Нет

Процедура выбора модели

- Проанализировать отличительные черты проекта по критериям, представленным в виде вопросов таблиц 1-4
- Ответить на вопросы по анализируемому проекту, отметив слова «да» или «нет» в соответствующих строках таблиц 1-4. Если слов «да» или «нет» в строке несколько, необходимо отметить все из них (все «да» или все «нет»)
- Расположить по степени важности категории (таблицы) и/или критерии, относящиеся к каждой категории (вопросы внутри таблиц), относительно проекта, для которого выбирается модель ЖЦ
- Выбрать ту модель, которая соответствует столбцу с наибольшим количеством отмеченных ответов с учетом их степени важности (с наибольшим количеством отмеченных ответов в верхней части приоритетных таблиц)

Подгонка модели жизненного цикла разработки ПО

- Ознакомьтесь с различными моделями
- Просмотрите и проанализируйте возможные виды работ: разработка, модернизация, сопровождение и т.д.
- Выберите самый подходящий жизненный цикл, используя для этого матрицы критериев: высокая степень риска, пользовательский интерфейс, высокая надежность, время поставки на рынок/выпуска продукта, приоритеты пользователя, уточнение требований, ожидаемый срок эксплуатации системы, технология, размер и сложность, возможный параллелизм, а также интерфейсы для существующих и новых систем
- Проанализируйте, насколько выбранный жизненный цикл соответствует стандартам вашей организации, ваших заказчиков или типа проекта — ISO, IEEE и т.д.
- Сформулируйте набор фаз и действий, образующих каждую фазу
- Определите внутренние и внешние производимые продукты
- Определите шаблоны и внутреннее содержимое поставляемых продуктов
- Определите действия по обзору, инспектированию, верификации и аттестации, а также стадии проекта
- Выполните оценку эффективности схемы жизненного цикла и проведите ее модернизацию там, где это необходимо

Пример выбора модели ЖЦ

Постановка задачи: В связи с переходом на новую программную платформу логистическая компания планирует внести изменения в свою информационную систему в части баз данных и учета складских запасов. Остальные модули программы останутся неизменными. Для выполнения этой задачи планируется привлечь команду высокопрофессиональных программистов из 6 человек, которые разрабатывали предыдущую версию системы. Какую модель жизненного цикла можно было бы применить при разработке системы? Обоснуйте свой ответ – приведите не менее 5-и аргументов, подтверждающих правильность Вашего выбора

Ответ: **каскадная** модель жизненного цикла

1. требования к системе хорошо известны и стабильны
2. проблемы предметной области проекта не являются новыми для большинства разработчиков
3. присутствие пользователей в процессе разработки не предусматривается
4. проект не определяет новое направление продукции для организации
5. интерфейсные модули остаются практически неизменными