**Лабораторная работа №3**

Тема: Выбор модели, метода и подхода разработки программы.

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| № критерия | Критерии категории требований | Каскадная | V-образная | RAD | Инкрементная | Быстрого прототипирования | Эволюционная |
| 1. | Являются ли требования к проекту легко определимыми и реализуемыми? | Да | Да | Да | Нет | Нет | Нет |
| 2. | Могут ли требования быть сформулированы в начале ЖЦ? | Да | Да | Да | Да | Нет | Нет |
| 3. | Часто ли будут изменяться требования на протяжении ЖЦ? | Нет | Нет | Нет | Нет | Да | Да |
| 4. | Нужно ли демонстрировать требования с целью их определения? | Нет | Нет | Да | Нет | Да | Да |
| 5. | Требуется ли проверка концепции программного средства или системы? | Нет | Нет | Да | Нет | Да | Да |
| 6. | Будут ли требования изменяться или уточняться с ростом сложности системы (программного средства) в ЖЦ? | Нет | Нет | Нет | Да | Да | Да |
| 7. | Нужно ли реализовать основные требования на ранних этапах разработки? | Нет | Нет | Да | Да | Да | Да |

Таблица 3 **–** Выбор модели жизненного цикла на основе характеристик требований

Вычисления: за каскадную – 2, за V – образную – 2, RAD – 5, инкрементная – 3, быстрого прототипирования – 5, эволюционная – 5.

Итог: на основе результатов заполнения табл. 3 подходящей является Инкремнтная, Быстрого прототипирования и эволюционная.

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| № критерия | Критерии категории команды разработчиков  проекта | Каскадная | V-образная | RAD | Инкрементная | Быстрого прототипирования | Эволюционная |
| 1. | Являются ли проблемы предметной области проекта новыми для большинства разработчиков? | Нет | Нет | Нет | Нет | Да | Да |
| 2. | Являются ли инструментальные средства, используемые в проекте, новыми для большинства разработчиков? | Да | Да | Нет | Нет | Нет | Да |
| 3. | Изменяются ли роли участников проекта на протяжении ЖЦ? | Нет | Нет | Нет | Да | Да | Да |
| 4. | Является ли структура процесса разработки более значимой для разработчиков, чем гибкость? | Да | Да | Нет | Да | Нет | Нет |
| 5. | Важна ли легкость распределения человеческих ресурсов проекта? | Да | Да | Да | Да | Нет | Нет |
| 6. | Приемлет ли команда разработчиков оценки, проверки, стадии разработки? | Да | Да | Нет | Да | Да | Да |

Вычисления: за каскадную – 4, за V – образную – 4, RAD – 1, инкрементная – 4, быстрого прототипирования – 3, эволюционная – 4.

Итог: на основе результатов заполнения табл. 4 подходящей является инкрементная модель, каскадная и эволюционная.

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| № критерия | Критерии категории коллектива пользователей | Каскадная | V-образная | RAD | Инкрементная | Быстрого прототипирования | Эволюционная |
| 1. | Будет ли присутствие пользователей ограничено в ЖЦ разработки? | Да | Да | Нет | Да | Нет | Да |
| 2. | Будут ли пользователи оценивать текущее состояние программного продукта (системы) в процессе разработки? | Нет | Нет | Нет | Да | Да | Да |
| 3. | Будут ли пользователи вовлечены во все фазы ЖЦ разработки? | Нет | Нет | Да | Нет | Да | Нет |
| 4. | Будет ли заказчик отслеживать ход выполнения проекта? | Нет | Нет | Нет | Нет | Да | Да |

Вычисления: за каскадную – 1, за V – образную – 1, RAD – 1, инкрементная – 2, быстрого прототипирования – 3, эволюционная – 3.

Итог: на основе результатов заполнения табл. 5 подходящей является модель быстрого проектирования и эволюционная.

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| № критерия | Критерии категории типов проекта и рисков | Каскадная | V-образная | RAD | Инкрементная | Быстрого прототипирования | Эволюционная |
| 1. | Разрабатывается ли в проекте продукт нового для организации направления? | Нет | Нет | Нет | Да | Да | Да |
| 2. | Будет ли проект являться расширением существующей системы? | Да | Да | Да | Да | Нет | Нет |
| 3. | Будет ли проект крупно- или среднемасштабным? | Нет | Нет | Нет | Да | Да | Да |
| 4. | Ожидается ли длительная эксплуатация продукта? | Да | Да | Нет | Да | Нет | Да |
| 5. | Необходим ли высокий уровень надежности продукта проекта? | Нет | Да | Нет | Да | Нет | Да |
| 6. | Предполагается ли эволюция продукта проекта в течение ЖЦ? | Нет | Нет | Нет | Да | Да | Да |
| 7. | Велика ли вероятность изменения системы (продукта) на этапе сопровождения? | Нет | Нет | Нет | Да | Да | Да |
| 8. | Является ли график сжатым? | Нет | Нет | Да | Да | Да | Да |
| 9. | Предполагается ли повторное использование компонентов? | Нет | Нет | Да | Да | Да | Да |
| 10. | Являются ли достаточными ресурсы (время, деньги, инструменты, персонал)? | Нет | Нет | Нет | Нет | Да | Да |

Вычисления: за каскадную – 2, за V – образную – 3, RAD – 3, инкрементная – 9, быстрого прототипирования – 7, эволюционная – 9.

Итог: на основе результатов заполнения табл. 5 подходящей является модель быстрого проектирования, инкрементная, эволюционная, инкрементная и RAD.

Общие вычисления: за каскадную – 9, за V – образную – 10, RAD – 10, инкрементная – 18, быстрого прототипирования – 19, эволюционная – 21.

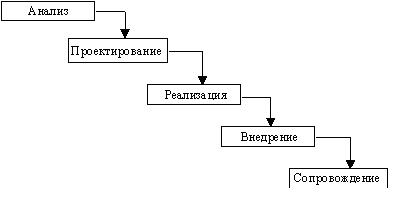
Общий итог: на основе заполнения таблиц 3 – 6 наиболее подходящей является эволюционная модель.

Каскадная модель жизненного цикла разработки ПО

**Каскадная (водопад)**

Водопадная или каскадная модель разработки программного обеспечения (waterfall, водопад) — это процесс разработки, в котором последовательно проходят фазы сбора и анализа требований, проектирования и прототипирования, реализации, тестирования, интеграции и поддержки.

Следуя водопадной модели, разработчики строго последовательно переходят от одного процесса к другому. Сначала полностью завершается этап сбора требований — составлется список выявленных требований к ПО. Только после того как требования собраны и зафиксированы, происходит переход к процессу проектирования, в ходе которого формируется техническое задание на разработку, содержащее описание задач, стоящих перед разработчиками, а также способов их реализации и имеющихся ограничений. После того как проектирование полностью завершено, разработчиками выполняется реализация проекта согласно техническому заданию — если на примере разработки сайта, то рисуется дизайн, осуществляется верстка и программирование. После того как фаза разработки завершена, производится тестирование и отладка разработанного продукта: на этой стадии выявляются и устраняются все недочёты согласно техническому заданию. После этого осуществляется запуск проекта и обеспечивается его поддержка — обеспечение работоспособности и устранение ошибок.



Положительные стороны применения каскадного подхода заключаются в следующем:

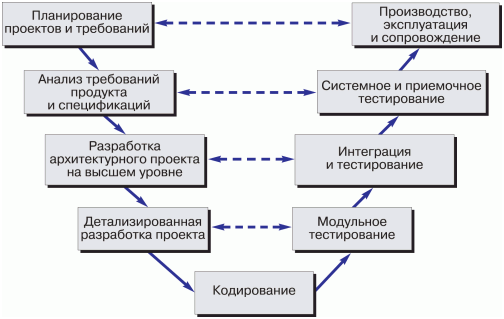
* на каждом этапе формируется законченный набор проектной документации, отве-чающий критериям полноты и согласованности;
* выполняемые в логичной последовательности этапы работ позволяют планировать сроки завершения всех работ и соответствующие затраты.

**V – образная**

V-модель – это улучшенная версия классической каскадной модели. Здесь на каждом этапе происходит контроль текущего процесса, для того чтобы убедится в возможности перехода на следующий уровень. В этой модели тестирование начинается еще со стадии написания требований, причем для каждого последующего этапа предусмотрен свой уровень тестового покрытия.

Для каждого уровня тестирования разрабатывается отдельный тест-план, то есть во время тестирования текущего уровня, мы также занимаемся разработкой стратегии тестирования следующего. Создавая тест-планы, мы также определяем ожидаемые результаты тестирования и указываем критерии входа и выхода для каждого этапа.

В V-модели каждому этапу проектирования и разработки системы соответствует отдельный уровень тестирования. Здесь процесс разработки представлен нисходящей последовательностью в левой части условной буквы V, а стадии тестирования – на ее правом ребре. Соответствие этапов разработки и тестирования показано горизонтальными линиями.



Плюсы и минусы V-модели:

+ строгая этапизация;

+ планирование тестирования и верификация системы производятся на ранних этапах;

+ улучшенный, по сравнению с каскадной моделью, тайм-менеджмент;

+ промежуточное тестирование.

— недостаточная гибкость модели;

— собственно создание программы происходит на этапе написания кода, то есть уже в середине процесса разработки;

— недостаточный анализ рисков;

— нет работы с параллельными событиями и возможности динамического внесения изменений.

Когда использовать V-модель:

– В проектах, в которых существуют временные и финансовые ограничения;

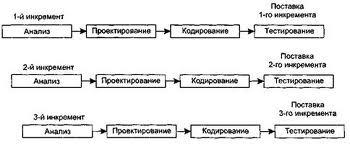
– Для задач, которые предполагают более широкое, по сравнению с каскадной моделью, тестовое покрытие.

Инкрементная модель жизненного цикла разработки ПО

**Инкрементная**

Инкрементная модель — это метод, в котором проект проектируется, реализуется и тестируется инкрементно (то есть каждый раз с небольшими добавлениями) до самого окончания разработки. Это включает в себя как разработку, так и дальнейшую поддержку продукта. Он считается законченным в то время, когда удовлетворяет всем требованиям. Модель объединяет элементы каскадной модели с прототипированием.

Проект можно разложить на несколько составляющих, каждая из которых спроектирована и построена независимо от другой (билд). Каждый такой компонент поставляется клиенту по мере готовности, что позволяет сразу начать использовать продукт и избежать длительной разработки. Также это стимулирует большие инвестиционные затраты, но сокращает время ожидания результата. Модель помогает «сгладить углы», вместо того, чтобы выкатывать пользователю совершенно новую систему разом.



### Преимущества инкрементной модели

* Рабочее приложение выходит на ранней стадии жизненного цикла продукта
* Гибкость. Изменить масштабы и требования проекта относительно менее затратно
* Небольшие итерации упрощают тестирование и внесение правок
* Проще идентифицировать риски, справиться с ними
* Каждая итерация — простая в управлении контрольная точка проекта

### Недостатки инкрементной модели

* Каждая фаза итерации неподвижна
* Могут возникнуть проблемы относительно архитектуры системы, так как не все требования собраны заранее для всего жизненного цикла ПО

### Когда использовать инкрементную модель

* Инкрементная модель используется вкупе с четкими и понятными требованиями, которые внедряются пофазово
* Востребована при разработке веб-приложений и продуктов компаний-брендов.

**RAD**

RAD — модель быстрой разработки приложений, ключевыми для которой является скорость и удобство программирования.

Данная модель, исходя из особенностей ее реализации и целей ее использования, может поддерживать как инкрементную, так и эволюционную стратегию разработки программного обеспечения. Как правило, RAD-модели используются в составе другой модели для ускорения цикла разработки прототипа системы или программного средства. При невысокой сложности проектов RAD-модели могут применяться как независимые.

Модели жизненного цикла, реализующие инкрементную или эволюционную стратегию разработки, широко применяют понятие быстрого прототипирования. RAD-модель представляет собой модель, на использовании которой базируется прототипирование.

Характерной чертой RAD-модели является короткое время перехода от анализа требований до создания полной системы или программного средства. Подход RAD предполагает наличие:



• небольших групп профессиональных разработчиков (до 7 чел.), выполняющих работы по проектированию отдельных подсистем программного обеспечения;

• хорошо проработанного производственного графика;

• циклической реализации в программном продукте требований, получаемых в результате взаимодействия с заказчиком. Жизненный цикл программного обеспечения в соответствии с

При использовании RAD-модели в соответствующем ей проекте

проявляются следующие ее достоинства:

• сокращение продолжительности цикла разработки и всего проекта в целом;

• сокращение количества разработчиков;

• вследствие предыдущих факторов — снижение стоимости проекта (также и за счет использования мощных инструментальных средств);

• сокращение риска несоблюдения графика;

• сокращение риска, связанного с неудовлетворенностью заказчика полученным программным продуктом, за счет привлечения его к циклу разработки;

• возможность повторного использования разработанных компонентов; это достоинство проявляется при использовании RAD-mo- дели в составе инкрементной или эволюционной модели. В этом случае наращивание функциональных возможностей осуществляется на базе разработанных ранее компонентов.

Основными недостатками RAD-модели при использовании в неподходящем для нее проекте являются:

• необходимость постоянного участия пользователей в процессе разработки, что не всегда выполнимо и может повлиять на качество конечного продукта;

• жесткость временных ограничений на разработку прототипа;

• сложность определения и ограничения затрат и сроков завершения работы над продуктом.

RAD-модель может эффективно применяться в следующих случаях:

• при разработке программных продуктов, если они поддаются моделированию, являются некритическими, имеют небольшой размер и низкую производительность, относятся к известной разработчикам предметной области, являются информационными системами;

• требования для программного обеспечения хорошо известны;

• имеются пригодные к повторному использованию в разрабатываемом ПО компоненты;

• пользователь может принимать постоянное участие в процессе разработки;

• в проекте заняты разработчики, обладающие достаточными навыками в использовании инструментальных средств разработки;

• при разработке ПП, для которых требуется быстрое наращивание функциональных возможностей;

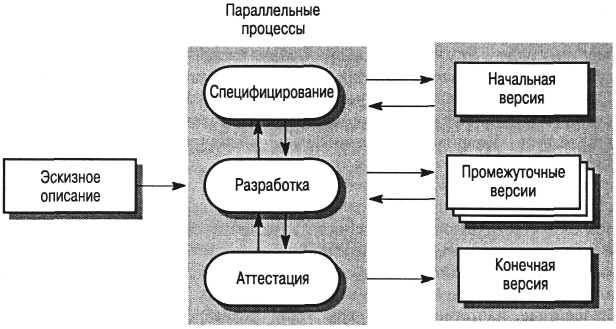
• при невысокой степени технических рисков;

• в составе других моделей жизненного цикла.

Эволюционная модель жизненного цикла ПО

**Эволюционная модель**

Эта модель основана на следующей идее: разрабатывается первоначальная версия программного продукта, которая передается на испытание пользователям, затем она дорабатывается с учетом мнения пользователей, получается промежуточная версия продукта, которая также проходит "испытание пользователем", снова дорабатывается и так несколько раз, пока не будет получен необходимый программный продукт



Эволюционный подход часто более эффективен, чем подход, построенный на основе каскадной модели, особенно если требования заказчика могут меняться в процессе разработки системы. Достоинством процесса создания ПО, построенного на основе эволюционного подхода, является то, что спецификация может разрабатываться постепенно, по мере того как заказчик (или пользователи) осознает и сформулирует те задачи, которые должно решать программное обеспечение.

Вместе с тем данный подход имеет и некоторые недостатки.

1. Многие этапы процесса создания ПО не документированы. Менеджерам проекта создания ПО необходимо регулярно документально отслеживать выполнение работ. Но если система разрабатывается быстро, то экономически не выгодно документировать каждую версию системы.

2. Система часто получается плохо структурированной. Постоянные изменения в требованиях приводят к ошибкам и упущениям в структуре ПО. Со временем внесение изменений в систему становится все более сложным и затратным.

3. Часто требуются специальные средства и технологии разработки ПО. Это вызвано необходимостью быстрой разработки версий программного продукта. Но, с другой стороны, это может привести к несовместимости некоторых применяемых средств и технологий, что, в свою очередь, требует наличия в команде разработчиков специалистов высокого уровня.

# Модель быстрого прототипирования

Модель быстрого протитипирования предназначена для быстрого создания прототипов продукта с целью уточнения требований и поэтапного развития прототипов в конечный продукт. Скорость (высокая производительность) выполнения проекта обеспечивается планированием разработки прототипов и участием заказчика в процессе разработки.

Начало жизненного цикла разработки помещено в центре эллипса. Совместно с пользователем разрабатывается предварительный план проекта на основе предварительных требований. Результат начального планирования - документ, опи­сывающий в общих чертах примерные графики и результативные данные.



Модель протипирования обладает целым рядом преимуществ:

1) Взаимодействие заказчика с разрабатываемой системой начинается на раннем этапе;

2) Благодаря реакции заказчика на прототип сводится к минимуму число неточностей в требованиях;

3) Снижается вероятность возникновения путаницы, искажения информации или недоразумений при определении требований к программному прдукту, что приводит к созданию более качественного программного продукта;

4) В процессе разработки всегда можно учесть новые, даже неожиданные требования заказчика;

5) Прототип представляет собой формальную спецификацию, воплощенную в программный продукт;

6) Прототип позволяет очень гибко выполнять проектирование и разработку, включая несколько итераций на всех фазах жизненного цикла разработки;

7) Заказчик всегда видит прогресс в процессе разработки программного продукта;

8) Возможность возникновения противоречий между разработчиками и заказчиками сведена к минимуму;

9) Уменьшается число доработок, что снижает стоимость разработки: возникающие проблемы решаются на ранних стадиях, что резко сокращает расходы на их устранение; заказчики принимают участие в процессе разработки на протяжении всего жизненного цикла и в конечном итоге в большей степени довольны результатом работы.

Кроме указанных достоинств модели прототипирования присущ и целый ряд недостатков:

1) Решение сложных задач может отодвигаться на будущее;

2) Заказчик может предпочесть получить прототип, а не законченную полную версию программного продукта;

3) Прототипирование может неоправданно затянуться;

4) Перед началом работы неизвестно, сколько итераций придется выполнить.

Модель прототипирования рекомендуется применять в следующих случаях:

1) Требования к программному продукту заранее неизвестны;

2) Требования не постоянны или неудачно сформулированы;

3) Требования необходимо уточнить;

4) Нужна проверка концепции;

5) Существует потребность в пользовательском интерфейсе;

6) Выполняется новая, не имеющая аналогов разработка;

7) Разработчики не уверены в том, какое решение следует выбрать.