**Лабораторная работа №3.**

**Тема:** Выбор модели, метода и подхода разработки программы.

**Цель работы:**

1. Изучить основные стратегии разработки и модели жизненного цикла.
2. Научиться осуществлять выбор стратегии и модели жизненного цикла для разработки конкретного проекта и обосновывать свой выбор.

***Задание №1. Изучение моделей ЖЦ.***

Провести анализ изученных вами стратегий разработки и моделей жизненного цикла. (дать краткую характеристику каждой модели, описать основные преимущества, недостатки, для решения каких задач может использоваться). Изученную информацию по каждой модели представить в конспекте в виде ответов на контрольные вопросы.

1. **Каскадная модель (водопад) и каскадная модель с возвратом.**

В этой модели разработка осуществляется поэтапно: каждая следующая стадия начинается только после того, как заканчивается предыдущая. Если всё делать правильно, «водопад» будет наиболее быстрой и простой моделью. Применяется уже почти полвека, с 1970-х.

Преимущества «водопада»

* Разработку просто контролировать. Заказчик всегда знает, чем сейчас заняты программисты, может управлять сроками и стоимостью.
* Стоимость проекта определяется на начальном этапе. Все шаги запланированы уже на этапе согласования договора, ПО пишется непрерывно «от и до».
* Не нужно нанимать тестировщиков с серьёзной технической подготовкой. Тестировщики смогут опираться на подробную техническую документацию.

Недостатки каскадной модели

* Тестирование начинается на последних этапах разработки. Если в требованиях к продукту была допущена ошибка, то исправить её будет стоить дорого. Тестировщики обнаружат её, когда разработчик уже написал код, а технические писатели — документацию.
* Заказчик видит готовый продукт в конце разработки и только тогда может дать обратную связь. Велика вероятность, что результат его не устроит.
* Разработчики пишут много технической документации, что задерживает работы. Чем обширнее документация у проекта, тем больше изменений нужно вносить и дольше их согласовывать.

«Водопад» подходит для разработки проектов в медицинской и космической отрасли, где уже сформирована обширная база документов (СНиПов и спецификаций), на основе которых можно написать требования к новому ПО.

При работе с каскадной моделью основная задача — написать подробные требования к разработке. На этапе тестирования не должно выясниться, что в них есть ошибка, которая влияет на весь продукт.

1. **Спиральная модель.**

Используя эту модель, заказчик и команда разработчиков серьёзно анализируют риски проекта и выполняют его итерациями. Последующая стадия основывается на предыдущей, а в конце каждого витка — цикла итераций — принимается решение, продолжать ли проект. Эту модель начали использовать в 1988 году.

Рассмотрим, как функционирует эта модель, на примере разработки системы «Умный дом».

1. Заказчик решил, что хочет сделать такую систему, и заказал программистам реализовать управление чайником с телефона. Они начали действовать по модели «водопад»: выслушали идею, провели анализ предложений на рынке, обсудили с заказчиком архитектуру системы, решили, как будут её реализовывать, разработали, протестировали и «выкатили» конечный продукт.
2. Заказчик оценил результат и риски: насколько нужна пользователям следующая версия продукта — уже с управлением телевизором. Рассчитал сроки, бюджет и заказал разработку. Программисты действовали по каскадной модели и представили заказчику более сложный продукт, разработанный на базе первого.
3. Заказчик подумал, что пора создать функциональность для управления холодильником с телефона. Но, анализируя риски, понял, что в холодильник сложно встроить Wi-Fi-модуль, да и производители не заинтересованы в сотрудничестве по этому вопросу. Следовательно, риски превышают потенциальную выгоду. На основе полученных данных заказчик решил прекратить разработку и совершенствовать имеющуюся функциональность, чтобы со временем понять, как развивать систему «Умный дом».

Спиральная модель похожа на инкрементную, но здесь гораздо больше времени уделяется оценке рисков. С каждым новым витком спирали процесс усложняется. Эта модель часто используется в *исследовательских проектах и там, где высоки* риски.

Преимущества спиральной модели

* Большое внимание уделяется проработке рисков.

Недостатки спиральной модели

* Есть риск застрять на начальном этапе— бесконечно совершенствовать первую версию продукта и не продвинуться к следующим.
* Разработка длится долго и стоит дорого.

**3. V-образная модель.**

Это усовершенствованная каскадная модель, в которой заказчик с командой программистов одновременно составляют требования к системе и описывают, как будут тестировать её на каждом этапе. История этой модели начинается в 1980-х.



Преимущества V-образной модели

* Количество ошибок в архитектуре ПО сводится к минимуму.
* Недостатки V-образной модели
* Если при разработке архитектуры была допущена ошибка, то вернуться и исправить её будет стоить дорого, как и в «водопаде».

V-модель подходит для проектов, в которых важна надёжность и цена ошибки очень высока. Например, при разработке подушек безопасности для автомобилей или систем наблюдения за пациентами в клиниках.

1. **Модель быстрой разработки приложений (RAD-модель)**

RAD-модель обеспечивает экстремально короткий цикл разработки. RAD —высокоскоростная адаптация линейной последовательной модели, в которой быстрая разработка достигается за счет использования компонентно-ориентированного конструирования. Если требования полностью определены, а проектная область ограничена, RAD-процесс позволяет создать полностью функциональную систему за очень короткое время (60-90 дней).

Применение RAD возможно в том случае, когда каждая главная функция может быть завершена за 3 месяца. Каждая главная функция адресуется отдельной группе разработчиков, а затем интегрируется в целую систему.

Применение RAD имеет- и свои недостатки, и ограничения.

* Для больших проектов в RAD требуются существенные людские ресурсы (необходимо создать достаточное количество групп).
* RAD применима только для таких приложений, которые могут декомпозироваться на отдельные модули и в которых производительность не является критической величиной.
* RAD не применима в условиях высоких технических рисков (то есть при использовании новой технологии).

**5. Многопроходная модель.**

Многопроходная модель (рис.8) - это несколько итераций (повторное применение какой-либо операции) процесса построения прототипа ПП с добавлением на каждой следующей итерации новых функциональных возможностей или повышением эффективности ПП.

Предполагается, что на ранних этапах жизненного цикла разработки (планирование, анализ требований и разработка проектa выполняется конструирование ПП в целом. Тогда же определяется и число необходимых инкрементов и относящихся к ним функций. Каждый инкремент затем проходит через оставшиеся фазы жизненного цикла (кодирование и тестирование). Сначала выполняются конструирование, тестирование и реализация базовых функций, составляющих основу ПП. Последующие итерации направлены на улучшение функциональных возможностей ПП.

Преимущества многопроходной модели:

* в начале разработки требуются средства только для разработки и реализации основных функций ПП;
* после каждого инкремента получается функциональный продукт;
* снижается риск неудачи и изменения требований;
* улучшается понимание как разработчиками, так и пользователями ПП требований для более поздних итераций;
* инкременты функциональных возможностей легко поддаются тестированию.

Недостатки многопроходной модели:

* не предусмотрены итерации внутри каждого инкремента;
* определение полной функциональности должно быть осуще­ствлено в самом начале жизненного цикла разработки;
* может возникнуть тенденция оттягивания решения трудных задач;
* общие затраты на создание ПП не будут снижены по сравне­нию с другими моделями;
* обязательным условием является наличие хорошего планирования и проектирования.

Многопроходная модель может быть применена, если большинство требований к ПП будут сформулированы заранее, а для выполнения проекта будет выделен большой период времени.

**6. Модель прототипирования.**

Модель прототипитования позволяет создать прототип программного продукта до или в течение этапа составления требований к программному продукту. Потенциальные пользователи работают с этим прототипом, определяя его сильные и слабые стороны, о результатах сообщают разработчикам программного продукта. Таким образом, обеспечивается обратная связь между пользователями и разработчиками, которая используется для изменения или корректировки спецификации требований к программному продукту. В результате такой работы продукт будет отражать реальные потребности пользователей.

Жизненный цикл разработки программного продукта начинается с разработки плана проекта, затем выполняется быстрый анализ, после чего создаются база данных, пользовательский интерфейс и выполняется разработка необходимых функций. В результате этой работы получается документ, содержащий частичную спецификацию требований к программному продукту. Данный документ в дальнейшем является основой для итерационного цикла быстрого прототипирования.

В результате прототипирования разработчик демонстрирует пользователям готовый прототип, а пользователи оценивают его функционирование. После этого определяются проблемы, над устранением которых совместно работают пользователи и разработчики. Этот процесс продолжается до тех пор, пока пользователи не будут удовлетворены степенью соответствия программного продукта, поставленным перед ним требованиям. Затем прототип демонстрируют пользователям с целью получения предложений по его усовершенствованию, которые включаются в последовательные итерации до тех пор, пока рабочая модель не окажется удовлетворительной. После этого получают от пользователей официальное одобрение (утверждение) функциональных возможностей прототипа и выполняют его окончательное преобразование в готовый программный продукт.

***Задание 2. Выбор модели ЖЦ. Обосновать выбор моделей ЖЦ для разработки проекта в соответствии с вашим индивидуальным заданием.***

Выбор модели жизненного цикла на основе характеристик требований

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| № критерия | Критерии категории требований | Каскадная | V-образная | RAD | Инкре- ментная | Быстрого  прототипи- рования | Эволюци- онная |
| 1. | Являются ли требования к проекту легко определи-  мыми 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 оп-  ределения? | **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 изме- няться или уточняться с ростом сложности системы (программного средства) в  ЖЦ? | Нет | Нет | Нет | **Да** | **Да** | **Да** |
| 7. | Нужно ли реализовать ос-  новные требования на ран- них этапах разработки? | Нет | Нет | **Да** | **Да** | **Да** | **Да** |

###### Итоги

###### Каскадная 2

###### V-образная 2

###### RAD 1

###### Инкрементная 5

###### Быстрого прототипирования 5

###### Эволюционная 5

Выбор модели жизненного цикла на основе характеристики команды разработчиков

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| № критерия | Критерии категории команды разработчиков проекта | Каскадная | V-образная | RAD | Инкре- ментная | Быстрого  прототипи- рования | Эволюци- онная |
| 1. | Являются ли проблемы предметной области проек-  та новыми для большинст- ва разработчиков? | Нет | Нет | Нет | Нет | **Да** | **Да** |
| 2. | Являются ли инструмен- тальные средства, исполь- зуемые в проекте, новыми  для большинства разработ- чиков? | **Да** | **Да** | Нет | Нет | Нет | **Да** |
| 3. | Изменяются ли роли участ- ников проекта на протяже-  нии ЖЦ? | **Нет** | **Нет** | **Нет** | Да | Да | Да |
| 4. | Является ли структура про- цесса разработки более значимой для разработчи-  ков, чем гибкость? | Да | Да | **Нет** | Да | **Нет** | **Нет** |
| 5. | Важна ли легкость распре- деления человеческих ре-  сурсов проекта? | Да | Да | Да | Да | **Нет** | **Нет** |
| 6. | Приемлет ли команда раз-  работчиков оценки, про- верки, стадии разработки? | **Да** | **Да** | Нет | **Да** | **Да** | **Да** |

###### Итог:

###### Каскадная 3

###### V-образная 3

###### RAD - 2

###### Инкрементная 1

###### Быстрого прототипирования 4

###### Эволюционная 5

Выбор модели жизненного цикла

на основе характеристик коллектива пользователей

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| № критерия | Критерии категории коллектива пользователей | Каскадная | V-образная | RAD | Инкре- ментная | Быстрого  прототипи- рования | Эволюци- онная |
| 1. | Будет ли присутствие поль- зователей ограничено в ЖЦ  разработки? | **Да** | **Да** | Нет | **Да** | Нет | **Да** |
| 2. | Будут ли пользователи оце- нивать текущее состояние программного продукта (системы) в процессе раз- работки? | Нет | Нет | Нет | **Да** | **Да** | **Да** |
| 3. | Будут ли пользователи во-  влечены во все фазы ЖЦ разработки? | **Нет** | **Нет** | Да | **Нет** | Да | **Нет** |
| 4. | Будет ли заказчик отслежи- вать ход выполнения про- екта? | Нет | Нет | Нет | Нет | **Да** | **Да** |

###### Итог:

###### Каскадная 2

###### V-образная 2

###### RAD 0

###### Инкрементная 3

###### Быстрого прототипирования 2

###### Эволюционная 4

Выбор модели жизненного цикла на основе характеристик типа проектов и рисков

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| № криерия | Критерии категории типов проекта и рисков | Каскадная | V-образная | RAD | Инкре- ментная | Быстрого  прототипи- рования | Эволюци- онная |
| 1. | Разрабатывается ли в про- екте продукт нового для ор-  ганизации направления? | Нет | Нет | Нет | **Да** | **Да** | **Да** |
| 2. | Будет ли проект являться расширением существую- щей 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 про- дукта проекта? | **Нет** | Да | **Нет** | Да | **Нет** | Да |
| 6. | Предполагается ли эволю- ция продукта проекта в те-  чение ЖЦ? | Нет | Нет | Нет | **Да** | **Да** | **Да** |
| 7. | Велика ли вероятность из- менения системы (продук-  та) на этапе сопровожде- ния? | Нет | Нет | Нет | **Да** | **Да** | **Да** |
| 8. | Является ли график сжа-  тым? | Нет | Нет | **Да** | **Да** | **Да** | **Да** |
| 9. | Предполагается ли повтор- ное использование компо-  нентов? | Нет | Нет | **Да** | **Да** | **Да** | **Да** |
| 10. | Являются ли достаточными  ресурсы (время, деньги, ин- струменты, персонал)? | Нет | Нет | Нет | Нет | **Да** | **Да** |

###### Итог:

###### Каскадная 2

###### V-образная 1

###### RAD 3

###### Инкрементная 7

###### Быстрого прототипирования 9

###### Эволюционная 9

###### Итог:

###### Каскадная 9

###### V-образная 8

###### RAD 6

###### Инкрементная 16

###### Быстрого прототипирования 20

###### Эволюционная 23

Благодаря процедуре выбора модели ЖЦ ПС и систем Института SQI базируется на применении четырех таблиц вопросов, соответствующих предложенной данным институтом классификации проектов и выделенным на них ответам, соответствующие моему проекту, мы выбрали модель с наибольшим количеством отмеченных ответов. Это оказалась эволюционная модель.