**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:**

Модели жизненного цикла

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

Представляет собой однократный проход этапов разработки. Каждый этап разработки начинается после завершения предыдущего этапа. Возврат к уже выполненным этапам не предусматривается. Промежуточные продукты разработки в качестве версии программного средства (системы) не распространяются.

Использование данной стратегии наиболее эффективно в следующих случаях

1) при разработке проектов с четкими, неизменяемыми в течение ЖЦ требованиями и понятной реализацией;

2) при разработке проектов невысокой сложности, например:   
∙ создание программного средства или системы такого же типа, как уже разрабатывались разработчиками;

∙ создание новой версии уже существующего программного средства или системы;

∙ перенос уже существующего продукта на новую платформу;  
3) при выполнении больших проектов в качестве составной части моде лей ЖЦ, реализующих другие стратегии разработки

Реализовать классическую каскадную модель ЖЦ в чистом виде затруднительно ввиду сложности разработки ПС без возвратов к предыдущим шагам и изменения их результатов для устранения возникающих проблем.

1. *Спиральная модель.*Базовая концепция спиральной модели заключается в следующем. Каждый цикл разработки (итерация) представляет собой набор операций, соответствующий шагам в каскадной модели. Шагам каскадной модели соответствует и последовательность витков спиральной модели.

Данная модель состоит из фаз разработки концепции, анализа требований, проектирования системы/продукта, реализации (технического проектирования, программирования и сборки), сопровождения и расширения функциональных возможностей. Модель поделена на четыре квадранта. В каждый квадрант входят основные и вспомогательные действия по разработке продукта или системы. Особое внимание в модели уделяется анализу и путям разрешения рисков.

1. *V-образная модель*Основное назначение V-образной модели – обеспечение планирования тестирования (испытаний) системы и программного средства на ранних стадиях проекта.  
   V-образная модель поддерживает каскадную стратегию разработки. В ней выделены связи между шагами, предшествующими программированию, и соответствующими видами тестирования и испытаний. Данные связи увязывают деятельность по разработке планов испытаний и тестирования с деятельностью по подтверждению результатов соответствующих этапов. В V-образной модели возможна организация обратных связей между этапами модели.
2. *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.

1. *Многопроходная модель.*

Многопроходная модель – это несколько итераций процесса построения прототипа программного продукта с добавлением на каждой следующей итерации новых функциональных возможностей или повышением эффективности программного продукта.

Предполагается, что на ранних этапах жизненного цикла разработки (планирование, анализ требований и разработка проекта) выполняется конструирование программного продукта 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 программного продукта. Последующие итерации направлены на улучшение функциональных возможностей программного продукта.

1. *Модель прототипирования*.

Модель прототипирования 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.

**Задание 2:**

**Проектирование**

**2.1 Выбор стратегии разработки и модели жизненного цикла**

Для разработки веб-ресурса и мобильного приложения «Доставка китайской еды «Вэньсинь»» следует выбрать стратегию разработки и модель жизненного цикла. Осуществляем выбор посредством составления таблиц:

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

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| № 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 |

Вычисления: 3 за каскадную, 3 за V- образную, 6 за RAD, 4 за инкрементную, 4 за быстрого прототипирования и 4 за эволюционную.

Итог: На основе результатов заполнения табл. 3 подходящей является RAD модель.

Таблица 4 **–** Выбор модели жизненного цикла на основе характеристик команды разработчиков

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| № 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: 5 за каскадную, 5 за V-образную, 4 за RAD, 5 за инкрементную, 2 за быстрого прототипирования и 1 за эволюционную.

Итог: На основе результатов заполнения табл. 4 подходящими являются каскадная, V-образная и инкрементная модели.

Таблица 5 **–** Выбор модели жизненного цикла на основе характеристик коллектива пользователей

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| № 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: 0 за каскадную, 0 за V-образную, 2 за RAD, 1 за инкрементную, 4 за быстрого прототипирования и 2 за эволюционную.

Итог: На основе результатов заполнения табл. 5 подходящей является модель быстрого проектирования.

Таблица 6 **–** Выбор модели жизненного цикла на основе характеристик типа проектов и рисков

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| № 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: 4 за каскадную, 5 за V-образную, 3 за RAD, 7 за инкрементную, 5 за быстрого прототипирования и 7 за эволюционную.

Итог: На основе результатов заполнения табл. 6 подходящей является инкрементная и эволюционная модели.

Таблица 7 — общий итог.

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
|  | Каскадная | V-образная | RAD | Инкрементная | Быстрого прототипирования | Эволюционная |
| Выбор модели жизненного цикла на основе характеристик требований | 3 | 3 | 6 | 4 | 4 | 4 |
| Выбор модели жизненного цикла на основе характеристик команды разработчиков | 5 | 5 | 4 | 5 | 2 | 1 |
| Выбор модели жизненного цикла на основе характеристик коллектива пользователей | 0 | 0 | 2 | 1 | 4 | 2 |
| Выбор модели жизненного цикла на основе характеристик типа проектов и рисков | 4 | 5 | 3 | 7 | 5 | 7 |
| Итог: | 12 | 13 | 15 | 17 | 15 | 14 |

Общий итог: в итоге заполнения табл. 3 – 6 наиболее подходящей является инкрементная модель.

**Контрольные вопросы:**

1. **Модель – «кодирование – устранение ошибок»**

– поставить задачу;

– выполнить ее до успешного завершения либо отмены;

– проверить результат;

– повторить при необходимости с 1 шага.  
**Недостаток** данной модели – является невозможность возврата на предыдущие стадии. Считается, что каждая стадия представляет собой определенный этап, после завершения работ на котором, к нему больше не возвращаются.

**Каскадная модель.**Каскадная стратегия представляет собой однократный проход этапов разработки. Данная стратегия основана на полном определении всех требований к разрабатываемому программному средству или системе в начале процесса разработки. Каждый этап разработки начинается после завершения предыдущего этапа. Возврат к уже выполненным этапам не предусматривается. Промежуточные продукты разработки в качестве версии программного средства (системы) не распространяются.

Основными **достоинствами** каскадной стратегии, проявляемыми при разработке соответствующего ей проекта, являются:

1) стабильность требований в течение ЖЦ разработки;

2) необходимость только одного прохода этапов разработки, что обеспечивает простоту применения стратегии;

3) простота планирования, контроля и управления проектом;

4) доступность для понимания заказчиками.

К основным **недостаткам** каскадной стратегии, проявляемым при ее использовании в проекте, ей не соответствующем, следует отнести:

1) сложность полного формулирования требований в начале процесса разработки и невозможность их динамического изменения на протяжении ЖЦ;

2) линейность структуры процесса разработки; разрабатываемые ПС или системы обычно слишком велики и сложны, чтобы все работы по их созданию выполнять однократно; в результате возврат к предыдущим шагам для решения возникающих проблем приводит к увеличению финансовых затрат и нарушению графика работ;

3) непригодность промежуточных продуктов для использования;

4) недостаточное участие пользователя в процессе разработки ПС - только в самом начале (при разработке требований) и в конце (во время приемочных испытаний); это приводит к невозможности предварительной оценки пользователем качества программного средства или системы.

Области применения каскадной стратегии определяются ее достоинствами и ограничены ее недостатками. Использование данной стратегии наиболее эффективно в следующих случаях:

1) при разработке проектов с четкими, неизменяемыми в течение ЖЦ требованиями и понятной реализацией;

2) при разработке проектов невысокой сложности, например:

• создание программного средства или системы такого же типа, как уже разрабатывались разработчиками;

• создание новой версии уже существующего программного средства или системы;

• перенос уже существующего продукта на новую платформу;

3) при выполнении больших проектов в качестве составной части моделей ЖЦ, реализующих другие стратегии разработки.

**Инкрементная стратегия разработки программных средств и систем**

Инкрементная стратегия представляет собой многократный проход этапов разработки с запланированным улучшением результата.

Данная стратегия основана на полном определении всех требований к разрабатываемому программному средству (системе) в начале процесса разработки. Однако полный набор требований реализуется постепенно в соответствии с планом в последовательных циклах разработки.

Основными **достоинствами** инкрементной стратегии, проявляемыми при разработке соответствующего ей проекта, являются:

1) Возможность получения функционального продукта после реализации каждого инкремента;

2) Короткая продолжительность создания инкремента; это приводит к сокращению сроков начальной поставки, позволяет снизить затраты на первоначальную и последующие поставки программного продукта;

3) Предотвращение реализации громоздких спецификаций требований; стабильность требований во время создания определенного инкремента; возможность учета изменившихся требований;

4) Снижение рисков по сравнению с каскадной стратегией;

5) Включение в процесс пользователей, что позволяет оценить функциональные возможности продукта на более ранних этапах разработки и в конечном итоге приводит к повышению качества программного продукта, снижению затрат и времени на его разработку.

К основным **недостаткам** инкрементной стратегии, проявляющимся в результате ее несоответствующего применения, следует отнести:

1) Необходимость полного функционального определения системы или программного средства в начале ЖЦ для обеспечения планирования инкрементов и управления проектом;

2) Возможность текущего изменения требований к системе или программному средству, которые уже реализованы в предыдущих инкрементах;

3) Сложность планирования и распределения работ;

4) Проявление человеческого фактора, связанного с тенденцией к оттягиванию решения трудных проблем на поздние инкременты, что может нарушить график работ или снизить качество программного продукта.

Области применения инкрементной стратегии определяются ее достоинствами и ограничены ее недостатками. Использование данной стратегии наиболее эффективно в следующих случаях:

1) При разработке проектов, в которых большинство требований можно сформулировать заранее, но часть из них могут быть уточнены через определенный период времени;

2) При разработке сложных проектов с заранее сформулированными требованиями; для них разработка системы или программного средства за один цикл связана с большими трудностями;

3) При необходимости быстро поставить на рынок продукт, имеющий базовые функциональные свойства;

4) При разработке проектов с низкой или средней степенью рисков;

5) При выполнении проекта с применением новых технологий.

**Эволюционная стратегия разработки программных средств и систем**

Эволюционная стратегия представляет собой многократный проход этапов разработки. Данная стратегия основана на частичном определении требований к разрабатываемому программному средству или системе в начале процесса разработки. Требования постепенно уточняются в последовательных циклах разработки. Результат каждого цикла разработки обычно представляет собой очередную поставляемую версию программного средства или системы.

Следует отметить, что в общем случае для эволюционной стратегии характерно существенно меньшее количество циклов разработки при большей продолжительности цикла по сравнению с инкрементной стратегией. При этом результат каждого цикла разработки (очередная версия программного средства или системы) гораздо сильнее отличается от результата предыдущего цикла.

Основными **достоинствами** эволюционной стратегии*,* проявляемыми при разработке соответствующего ей проекта, являются:

1)    возможность уточнения и внесения новых требований в процессе раз­работки;

2)    пригодность промежуточного продукта для использования;

3)    возможность управления рисками;

4)     обеспечение широкого участия пользователя в проекте, начиная с ран­них этапов, что минимизирует возможность разногласий между заказчиками и разработчиками и обеспечивает создание продукта высокого качества;

5)    реализация преимуществ каскадной и инкрементной стратегий.

К**недостаткам** эволюционной стратегии, проявляемым при ее несоот­ветствующем выборе, следует отнести:

1)    неизвестность точного количества необходимых итераций и слож­ность определения критериев для продолжения процесса разработки на сле­дующей итерации; это может вызвать задержку реализации конечной версии системы или программного средства;

2)    сложность планирования и управления проектом;

3)    необходимость активного участия пользователей в проекте, что реаль­но не всегда осуществимо;

4)    необходимость в мощных инструментальных средствах и методах прототипирования;

5)    возможность отодвигания решения трудных проблем на последующие циклы, что может привести к несоответствию полученных продуктов требова­ниям заказчиков.

*Области применения эволюционной стратегии* определяются ее достоин­ствами и ограничены ее недостатками. Использование данной стратегии наибо­лее эффективно в следующих случаях:

1)    при разработке проектов, для которых требования слишком сложны, неизвестны заранее, непостоянны или требуют уточнения;

2)    при разработке сложных проектов, в том числе:

•  больших долгосрочных проектов;

•  проектов по созданию новых, не имеющих аналогов ПС или систем;

•  проектов со средней и высокой степенью рисков;

•  проектов, для которых нужна проверка концепции, демонстрация технической осуществимости или промежуточных продуктов;

3)    при разработке проектов, использующих новые технологии.

1. Модель жизненного цикла ПО — структура, определяющая последовательность выполнения и взаимосвязи процессов, действий и задач на протяжении жизненного цикла.

**Каскадная стратегия**

*Каскадная модель (водопад)*

Одна из самых старых, подразумевает последовательное прохождение стадий, каждая из которых должна завершиться полностью до начала следующей. В модели Waterfall легко управлять проектом. Благодаря её жесткости, разработка проходит быстро, стоимость и срок заранее определены. Но это палка о двух концах. Каскадная модель будет давать отличный результат только в проектах с четко и заранее определенными требованиями, и способами их реализации. Нет возможности сделать шаг назад, тестирование начинается только после того, как разработка завершена или почти завершена. Продукты, разработанные по данной модели без обоснованного ее выбора, могут иметь недочеты, о которых становится известно лишь в конце из-за строгой последовательности действий.

*Каскадная модель с возвратом.*

Данная модель является почти эквивалентной по алгоритму предыдущей модели, однако при этом имеет обратные связи с каждым этапом жизненного цикла, при этом порождает очень весомый недостаток: 10-ти кратное увеличение затрат на разработку. Относится к первой группе моделей.

*V-образная модель.*

Унаследовала структуру «шаг за шагом» от каскадной модели. V-образная модель применима к системам, которым особенно важно бесперебойное функционирование. Например, прикладные программы в клиниках для наблюдения за пациентами, интегрированное ПО для механизмов управления аварийными подушками безопасности в транспортных средствах и так далее. Особенностью модели можно считать то, что она направлена на тщательную проверку и тестирование продукта, находящегося уже на первоначальных стадиях проектирования. Стадия тестирования проводится одновременно с соответствующей стадией разработки, например, во время кодирования пишутся модульные тесты.

**Инкрементная стратегия**

*Инкрементная модель*

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.

*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).

**Эволюционная модель**

*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. Отличительной чертой данной модели является то, что процессы специфицирования, разработки и аттестации ПО выполняются параллельно при постоянном обмене информацией между ними.

*Спиральная модель*

Спиральная модель — это модель процесса разработки программного обеспечения с учетом рисков. Это комбинация модели водопада и итеративной модели. Spiral Model помогает внедрить элементы разработки программного обеспечения из нескольких моделей процессов для программного проекта на основе уникальных шаблонов рисков, обеспечивая эффективный процесс разработки.

Каждая фаза спиральной модели в разработке программного обеспечения начинается с определения цели проектирования и заканчивается тем, что клиент просматривает прогресс.