**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.

Практическая часть работы

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).

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

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.

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 обеспе-чивает 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 моде-

лей ЖЦ, реализующих другие стратегии разработки (см., например, модели, приведенные на рис. 2.5 и рис. 2.12).

1. Спиральная модель.

Спира́льная модель представляет собой [процесс разработки программного обеспечения](https://ru.wikipedia.org/wiki/%D0%9F%D1%80%D0%BE%D1%86%D0%B5%D1%81%D1%81_%D1%80%D0%B0%D0%B7%D1%80%D0%B0%D0%B1%D0%BE%D1%82%D0%BA%D0%B8_%D0%BF%D1%80%D0%BE%D0%B3%D1%80%D0%B0%D0%BC%D0%BC%D0%BD%D0%BE%D0%B3%D0%BE_%D0%BE%D0%B1%D0%B5%D1%81%D0%BF%D0%B5%D1%87%D0%B5%D0%BD%D0%B8%D1%8F), сочетающий в себе как итеративность, так и этапность.

При использовании спиральной модели  проявляются следующие преимущества:

* спиральная модель разрешает пользователям "увидеть" систему на ранних этапах;
* эта модель разрешает пользователям активно принимать участие при планирова­нии, анализе рисков, разработке, а также при выполнении оценочных действий;
* в модели предусмотрена возможность гибкого проектирования
* реализованы преимущества инкрементной модели, а именно выпуск инкрементов,
* здесь не ставится цель выполнить невозможное — довести конструкцию до совершенства;
* обратная связь по направлению от пользователей к разработчикам выполняется с высокой частотой и на ранних этапах модели, что обеспечивает создание нужного продукта высокого качества;
* повышается продуктивность благодаря использованию пригодных для повторного использования свойств;
* повышается вероятность предсказуемого поведения системы с помощью уточнения поставленных целей;
* при использовании спиральной модели не нужно распределять заранее все необходимые для выполнения проекта финансовые ресурсы;

**Недостатки спиральной модели**

* если проект имеет низкую степень риска или небольшие размеры, модель может оказаться дорогостоящей. Оценка рисков после прохождения каждой спирали связана с большими затратами;
* модель имеет усложненную структуру, поэтому может быть затруднено ее применение разработчиками, менеджерами и заказчиками;
* серьезная нужда в высокопрофессиональных знаниях для оценки рисков;
* спираль может продолжаться до бесконечности, поскольку каждая ответная реак­ция заказчика на созданную версию может порождать новый цикл, что отдаляет окончание работы над проектом (принятие общего решения о прекращении про­цесса разработки);
* большое количество промежуточных стадий может привести к необходимости в обработке внутренней дополнительной и внешней документации;
* использование модели может оказаться дорогостоящим и даже недопустимым по средствам, так как время, затраченное на планирование, повторное определение целей, выполнение анализа рисков и прототипирование, может быть чрезмерным;
* при выполнении действий на этапе вне процесса разработки возникает необходи­мость в переназначении разработчиков;
* могут возникнуть затруднения при определении целей и стадий, указывающих на готовность продолжать процесс разработки на следующей итерации.

Главная задача — как можно быстрее показать пользователям системы работоспособный продукт, тем самым активизируя процесс уточнения и дополнения требований.

1. V-образная модель.

**V**-Model (или VEE **модель**) является **моделью** разработки информационных систем (ИС), направленной на упрощение понимания сложностей, связанных с разработкой систем.

При использовании V-образной модели при разработке проекта, для которого она в достаточной мере подходит, обеспечивается несколько преимуществ:

* в модели предусмотрены аттестация и верификация всех внешних и внутренних полученных данных, а не только самого программного продукта;
* в V-образной модели определение требований выполняется перед разработкой проекта системы, а проектирование ПО — перед разработкой компонентов;
* модель определяет продукты, которые должны быть получены в результате про­цесса разработки, причем каждые полученные данные должны подвергаться тестированию;
* благодаря модели менеджеры проекта может отслеживать ход процесса разработ­ки, так как в данном случае вполне возможно воспользоваться временной шкалой, а завершение каждой фазы является контрольной точкой;
* модель проста в применении.

Недостатки V-образной модели

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

1. Модель быстрой разработки приложений (RAD-модель)

rapid **application** development — **быстрая разработка приложений**) — концепция организации технологического процесса **разработки** программных продуктов, ориентированная на максимально быстрое получение качественного результата в условиях сильных ограничений по срокам и бюджету и нечётко определённых требований к продукту.

**Преимущества модели RAD**

1. Быстрое развитие продукта.
2. Разработка многоразовых мелких компонентов.
3. Повторный обзор в процессе разработки.
4. Интеграция повторно используемых компонентов на начальном уровне, следовательно, экономит усилия, несмотря на то, что не добавляются более крупные модули.
5. Конструктивная реакция.

**Недостатки модели RAD**

1. Требуется много усилий для сбора всех требований на начальном этапе.
2. Навыки моделирования имеют много зависимостей.
3. Не подходит для малобюджетного проекта.

RAD-технология не является универсальной, её целесообразно применять лишь если проект отвечает всем или некоторым из условий:

1. Сжатые сроки. Требуется максимально быстро создать систему, отвечающую требованиям сегодняшнего дня. Увеличение сроков создаёт высокую вероятность настолько значительного изменения фундаментальных положений, регламентирующих автоматизируемую деятельность, что система морально устареет ещё до завершения проектирования.
2. Нечётко определённые и/или изменяющиеся по ходу разработки требования. Заказчик весьма приблизительно представляет себе работу будущего программного продукта и не может четко сформулировать все требования к ПО. Требования могут быть вообще не определены к началу проекта либо могут изменяться по ходу его выполнения.
3. Ограниченный бюджет при готовности участия заказчика в разработке. У заказчика нет средств на оплату работы большой команды проектировщиков и разработчиков в течение длительного времени, но имеется готовность выделить специалистов для постоянного непосредственного участия в разработке и оценки её текущего состояния.
4. Небольшие объёмы либо возможность разбиения проекта на функциональные компоненты. Если предполагаемая система велика, необходимо, чтобы её можно было разбить на небольшие части, каждая из которых обладает четкой функциональностью и минимально зависит от других. Они могут выпускаться последовательно или параллельно (в последнем случае привлекается несколько RAD-групп).
5. [Графический интерфейс пользователя](https://ru.wikipedia.org/wiki/%D0%93%D1%80%D0%B0%D1%84%D0%B8%D1%87%D0%B5%D1%81%D0%BA%D0%B8%D0%B9_%D0%B8%D0%BD%D1%82%D0%B5%D1%80%D1%84%D0%B5%D0%B9%D1%81_%D0%BF%D0%BE%D0%BB%D1%8C%D0%B7%D0%BE%D0%B2%D0%B0%D1%82%D0%B5%D0%BB%D1%8F) — важнейший или один из важнейших компонентов системы. Именно в создании интерфейса RAD-технология даёт наибольшие преимущества, так как интерфейс демонстрируется непосредственно на прототипе, причём достаточно скоро после начала проекта. Возможно даже прямо привлечь представителя заказчика к проектированию интерфейса в визуальном редакторе. Этот подход позволяет избежать типичной ситуации, когда интерфейс, описанный пользователем в требованиях (как правило, без учёта технологических ограничений) ведёт себя на практике совсем не так, как рассчитывал пользователь, хотя формально система полностью соответствует документированным требованиям.
6. Низкая вычислительная сложность. Обработка данных в проекте сводится к комбинированию типовых операций, все или большинство из которых уже реализованы в виде доступных библиотек. Оригинальных алгоритмов обработки данных либо вообще не требуется, либо они достаточно просты и могут быть реализованы быстро и без особых затруднений.
7. 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.

Преимущества многопроходной модели: в начале разработки требуются средства только для разработки и реализации основных функций программного продукта; после каждого инкремента получается функциональный продукт; снижается риск неудачи и изменения требований; улучшается понимание как разработчиками, так и пользователями программного продукта 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 программного продукта не будут снижены по сравнению с другими моделями; обязательным условием является наличие хорошего планирования и проектирования.

1. Модель прототипирования.

Прототипирование — это процесс построения рабочей модели системы. Прототип — это эквивалент экспериментальной модели или "макета" в мире аппаратного обеспечения.

**Преимущества структурной эволюционной модели быстрого прототипирования:**

* конечный пользователь может "увидеть" системные требования в процессе их сбора командой разработчиков; таким образом, взаимодействие заказчика с системой начинается на раннем этапе разработки;
* исходя из реакции заказчиков на демонстрации разрабатываемого продукта, разработчики получают сведения об одном или нескольких аспектах поведения системы, благодаря чему сводится к минимуму количество неточностей в требованиях;
* снижается возможность возникновения путаницы, искажения информации или недоразумений при определении системных требований, что несомненно приводит к созданию более качественного конечного продукта;
* модель представляет собой формальную спецификацию, воплощенную в рабочую модель;
* модель позволяет выполнять гибкое проектирование и разработку, включая несколько итераций на всех фазах жизненного цикла;
* благодаря меньшему объему доработок уменьшаются затраты на разработку;
* благодаря тому что проблема выявляется до привлечения дополнительных ресурсов сокращаются общие затраты;
* обеспечивается управление рисками;
* документация сконцентрирована на конечном продукте, а не на его разработке.

**Недостатки структурной эволюционной модели быстрого прототипирования:**

* с учетом создания рабочего прототипа, качеству всего ПО или долгосрочной эксплуатационной надежности может быть уделено недостаточно внимания.
* иногда в результате использования модели получают систему с низкой рабочей характеристикой, особенно если в процессе ее выполнения пропускается этап подгонки;
* при использовании модели решение трудных проблем может отодвигаться на будущее. В результате это приводит к тому, что последующие полученные продукты могут не оправдать надежды, которые возлагались на прототип;
* если пользователи не могут участвовать в проекте на итерационной фазе быстрого прототипирования жизненного цикла, на конечном продукте могут отразиться неблагоприятные воздействия, включая проблемы, связанные с его качественной характеристикой;
* на итерационном этапе прототипирования быстрый прототип представляет собой частичную систему. Если выполнение проекта завершается досрочно, у конечного пользователя останется только лишь частичная система;
* если язык или среда прототипирования не сочетаются с производственным языком или окружением, всесторонняя реализация продукционной системы может быть задержана;
* прототипирование вызывает зависимость и может продолжаться слишком долго. Нетренированные разработчики могут попасть в так называемый цикл "кодирование — устранение ошибок" (code-and-fix cycle), что приводит к дорогостоящим незапланированным итерациям прототипирования;
* структурные методы не используются, чтобы не помешать выполнению анализа. При прототипировании необходимо провести "реальный" анализ требований, осуществить проектирование и обратить внимание на качество с целью создания программы, допускающей сопровождение, точно так же, как и в любой другой модели жизненного цикла (хотя на эти действия может понадобиться меньше времени и ресурсов).

**Область применения структурной эволюционной модели быстрого прототипирования**: менеджер проекта может быть уверен в необходимости применения структурной эволюционной модели быстрого прототипирования, если:

* требования не известны заранее;
* требования не постоянны или могут быть неверно истолкованы или неудачно сформулированы;
* существует потребность в разработке пользовательских интерфейсов;
* осуществляются временные демонстрации;
* построенное по принципу структурной модели, эволюционное быстрое прототипирование можно успешно использовать в больших системах, в которых некоторые модели подвергаются прототипированию, а некоторые— разрабатываются более традиционным образом;
* требуется уменьшить неточности в определении требований; т.е. уменьшается риск создания системы, которая не имеет никакой ценности для заказчика;
* 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;
* задействованы высокотехнологические системы с интенсивным применением ПО, где можно лишь обобщенно, но не точно сформулировать требования, лежащие за пределами главной характеристики;
* разрабатывается ПО, особенно в случае программ, когда проявляется средняя и высокая степень риска;
* осуществляется применение в комбинации с каскадной моделью: на начальном этапе проекта используется прототипирование, а на последнем — фазы каскадной модели с целью обеспечения функциональной эффективности системы и качества;
* прототипирование всегда следует использовать вместе с элементами анализа и проектирования, применяемыми при объектно-ориентированной разработке. Быстрое прототипирование особенно хорошо подходит для разработки интенсивно используемых систем пользовательского интерфейса, таких как индикаторные панели для контрольных приборов, интерактивные системы, новые в своем роде продукты, а также системы обеспечения принятия решений, среди которых можно назвать подачу команд, управление или медицинскую диагностику.

Задание 2. Выбор модели ЖЦ.

Обосновать выбор моделей ЖЦ для разработки проекта в соответствии с вашим индивидуальным заданием.

**Рассматриваемая процедура состоит из следующей последовательности шагов:**

***1-й шаг****.* Проанализировать отличительные черты проекта по критериям категорий, представленным в виде вопросов.

***2-й шаг****.* Ответить на вопросы по анализируемому проекту, отметив слова «да» или «нет» в соответствующих строках табл. 3.1 – 3.4. Если слов «да» или «нет» в строке несколько, необходимо отметить все из них (все «да» или все «нет»).

В качестве примера в табл. 3.1 выделены варианты ответов для проекта разработки сложного и критичного программного средства, требования к которому заранее не известны и будут уточняться по ходу разработки.

***3-й шаг****.* Расположить по степени важности категории (таблицы) и/или критерии, относящиеся к каждой категории (вопросы внутри таблиц), относительно проекта, для которого выбирается модель ЖЦ.

***4-й шаг.***Выбрать из моделей (см. табл. 3.1 – 3.4) 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). Выбранная модель ЖЦ является наиболее приемлемой для анализируемого проекта.

Таблица 3.1 – 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 |

Продолжение таблицы 3.1

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 |
| 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 |

Каскадная – 2, V-образная – 2, RAD – 5, инкрементная – 3, быстрого прототипирования – 5, эволюционная – 5.

На основе результатов заполнения табл. 3.1 наиболее подходящей является RAD, прототипирования и эволюционная модель.

Таблица 3.2 – 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 |
|  | |  | | | | | | | | |

Каскадная – 4, V-образная – 4, RAD – 5, инкрементная – 4, быстрого прототипирования – 3, эволюционная – 2.

На основе результатов заполнения табл. 3.2 наиболее подходящими являются RAD модель.

Таблица 3.3 – 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 |

Продолжение таблицы 3.3

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 |
| 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 |

Каскадная – 1, V-образная – 1, RAD – 1, инкрементная – 2, быстрого прототипирования – 3, эволюционная – 3.

На основе результатов заполнения табл. 3.3 наиболее подходящими являются каскадная и эволюционная модели.

Таблица 3.4 – 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 |

Продолжение таблицы 3.4

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 |
| 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 |

Каскадная – 3, V-образная – 4, RAD – 2, инкрементная – 6, быстрого прототипирования – 6, эволюционная – 8.

На основе результатов заполнения табл. 3.4 наиболее подходящей является эволюционная модель.

Исходя из результатов заполнения табл. 3.1 – 3.4 (каскадная – 10, V-образная – 11, RAD – 13, инкрементная – 15, быстрого прототипирования – 17, эволюционная – 18) наиболее подходящей моделью, для разработки данного ПП, является эволюционная модель.