| *voenmeh* | МИНОБРНАУКИ РОССИИ  федеральное государственное бюджетное образовательное учреждение  высшего образования  **«Балтийский государственный технический университет «ВОЕНМЕХ» им. Д.Ф. Устинова»**  **(БГТУ «ВОЕНМЕХ» им. Д.Ф. Устинова»)** |
| --- | --- |
| БГТУ.СМК-Ф-4.2-К5-01 |

| Факультет |  | О |  | Естественнонаучный |
| --- | --- | --- | --- | --- |
|  |  | шифр |  | наименование |
| Кафедра |  | О7 |  | Информационные системы и программная инженерия |
|  |  | шифр |  | наименование |
| Дисциплина |  | Разработка пользовательского интерфейса | | |

ИНДИВИДУАЛЬНАЯ ПРАКТИЧЕСКАЯ РАБОТА №2

| Выполнили студенты группы | | | |  | И508Б |
| --- | --- | --- | --- | --- | --- |
| Кабиров К.Р. Попов Д.А. Кулышева В.С. Тихонов Д.Н. | | | | | |
| Фамилия И.О. | | | | | |
| **РУКОВОДИТЕЛЬ** | | | | | |
|  | |  |  | | |
| Фамилия И.О. Подпись | | | | | |
| Защита |  | | | |  |
| «\_\_\_\_\_» |  | | | | 2023 г. |

САНКТ-ПЕТЕРБУРГ

2023 г.

**ВВЕДЕНИЕ**

На основании выбранного в предыдущей работе варианта задания, выбрать и описать не менее 5 моделей или методологий жизненного цикла и обосновать наиболее подходящую для разработки данного программного обеспечения.

Были выбраны следующие модели и методологии:

* Scrum методология
* итерационная модель;
* инкрементная модель;
* гибкая методология AGILE;
* методология XP.

**Scrum методология**

Scrum – это методология гибкой разработки ПО, которая была разработана для улучшения эффективности работы команды и ускорения процесса разработки. Она предполагает разбиение проекта на короткие итерации, называемые спринтами, и акцентирует внимание на эффективной коммуникации внутри команды и с заказчиком.

Процесс работы Scrum-команды включает в себя следующие шаги:

1. Разработка бэклога продукта
2. Сбор команды Scrum-команда
3. Планирование коротких циклов (спринтов)
4. Scrum-митинг, или стендап
5. Scrum-доска
6. Подведение итогов спринта
7. Обзор результатов работы над проектом

Scrum-команда – единое целое. Каждый член команды выполняет свою роль и работает в тесном сотрудничестве с другими членами команды. Это позволяет команде быстро реагировать на изменения в проекте и достигать поставленных целей.

**Основные преимущества Scrum методологии:**

* гибкость: Scrum подходит для работы с меняющимися требованиями и быстро изменяющейся средой разработки;
* ускорение процесса разработки: Разбиение проекта на спринты позволяет быстрее получать результаты и своевременно их корректировать;
* лучшая коммуникация: Scrum акцентирует внимание на эффективной коммуникации внутри команды и с заказчиком, что способствует более ясному пониманию требований и улучшению командной работы;
* снижение рисков: Scrum помогает своевременно выявлять и решать проблемы, что снижает риски для проекта;

**Основные недостатки Scrum методологии:**

* необходимость опытных и дисциплинированных участников команды: Scrum требует определенного уровня квалификации и дисциплины участников команды, чтобы правильно следовать процессу и добиться максимальной эффективности;
* ограниченность применимости: Scrum не является универсальным решением для всех проектов, и может не подходить для сложных и больших проектов;
* необходимость постоянного взаимодействия с заказчиком: Scrum требует постоянной обратной связи с заказчиком, что может стать сложным в случае, если заказчик находится в другой часовой зоне или не может постоянно находиться на связи;
* в целом, Scrum является эффективным методом гибкой разработки ПО, который может помочь ускорить процесс разработки и повысить качество продукта, если применять его правильно и с учетом специфики проекта.

Scrum методология используется в различных областях, но чаще всего она применяется в разработке программного обеспечения и веб-разработке. Некоторые из компаний, использующих Scrum методологию, включают Google, Netflix, Microsoft, Spotify, Ericsson, Dell и Adobe.

**Итерационная модель**

Итерационная модель разработки ПО является методологией, которая заключается в последовательном выполнении повторяющихся итераций (циклов) разработки, каждая из которых проходит определенные этапы, такие как анализ требований, проектирование, разработка, тестирование и внедрение. Каждая итерация представляет собой миниатюрный проект со своим циклом жизни, включающий основные этапы разработки ПО.

Основной принцип итерационной модели заключается в том, что последующие итерации (циклы) строятся на основе результатов предыдущих итераций, что позволяет быстро реагировать на изменения требований заказчика и вносить соответствующие изменения в процесс разработки без значительных затрат.

Процесс разработки в итерационной модели может быть описан следующим образом:

1. Анализ требований и определение основных целей проекта
2. Проектирование и создание прототипов
3. Разработка и тестирование
4. Внедрение

В конце каждой итерации происходит оценка и анализ проделанной работы. Если цели и качество работы были достигнуты, то процесс переходит к следующей итерации. Если нет, то сначала происходит внесение корректировок и изменений, после чего цикл повторяется.

**К основным преимуществам итерационной модели относятся:**

* гибкость – возможность быстро реагировать на изменения требований заказчика и вносить соответствующие изменения в процесс разработки без значительных затрат;
* ранний и более частый доступ к первоначальному продукту - клиент или заказчик получает быстрый доступ к первым результатам работы, которые могут быть использованы для корректировки требований или плана разработки;
* высокое качество конечного продукта – так как каждая итерация проходит через все этапы жизненного цикла проекта разработки ПО, включая анализ требований, проектирование, разработку, тестирование и внедрение, то возможно более точное определение требований и соответствия конечного продукта этим требованиям.

**К недостаткам итерационной модели можно отнести:**

* дополнительные затраты на каждую итерацию - итерационный процесс приводит к дополнительным затратам на каждую итерацию, так как каждая итерация проходит все этапы жизненного цикла проекта;
* необходимость в активном участии заказчика на всех фазах проекта – для эффективной реализации итерационной модели необходимо активное участие заказчика на всех фазах проекта, что может стать проблемой, если заказчик находится в другом регионе или имеет ограниченное время;
* недостаток структуры – в итерационной модели не всегда можно легко получить общую картину проекта, так как каждая итерация может иметь разное направление, что затрудняет управление проектом и, как следствие, его контроль;
* нужно хорошее планирование - чтобы избежать перекрывания этапов разных итераций и ущерба для продукта, необходимо быть строго организованным и хорошо планировать каждую итерацию;
* не каждый проект подходит для итерационной модели - не все проекты подходят для итерационной разработки, и в некоторых случаях модель может быть более сложной, чем другие методики разработки ПО;

Нет фиксированного бюджета и сроков, а также нужна сильная вовлеченность Заказчика в процесс – для некоторых Заказчиков это неприемлемые условия сотрудничества с разработчиком, им лучше подойдет водопадная модель.

Использование итерационной модели снижает риски глобального провала и растраты всего бюджета, получение несинхронизированных ожиданий и ошибочного понимания процессов как клиентом, так и каждым участником команды разработки. Оно также дает возможность завершения разработки в конце любой итерации (в каскадной модели вы должны прежде завершить все этапы).

**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.

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.

Есть и проблемы. Например, каждый новый билд должен быть интегрирован с предыдущим во всех системах. Разборка на билды тоже не элементарна. Если есть несколько билдов, которые еще и необходимо разобрать, все превращается в модель «Компилируй и исправляй ошибки». Как бы то ни было, большое количество сборок означает малое количество изменений между ними.

**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, внесение правок и определение рисков;
* 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.

Инкрементная модель используется с четкими и понятными требованиями, когда внедрение происходит поэтапно. Модель востребована при разработке веб-приложений и продуктов компаний-брендов.

**Гибкая методология AGILE**

Основной [метрикой](https://ru.wikipedia.org/wiki/%D0%9C%D0%B5%D1%82%D1%80%D0%B8%D0%BA%D0%B0_%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) agile-методов является рабочий продукт, который выполняется итерациями в 2-3 недели с минимально необходимым числом задач. Отдавая предпочтение непосредственному общению, agile-методы уменьшают объём письменной документации. Подразумевается, что после каждой итерации продукт можно выпускать.

Agile Manifesto – свод правил, который содержит 4 основные идеи и 12 принципов без практических советов.

Основные идеи:

* люди и взаимодействие важнее процессов и инструментов;
* работающий продукт важнее исчерпывающей документации;
* сотрудничество с заказчиком важнее согласования условий контракта;
* готовность к изменениям важнее следования первоначальному плану.

Основополагающие принципы [Agile Manifesto](https://ru.wikipedia.org/wiki/Agile_Manifesto):

* наивысшим приоритетом признается удовлетворение заказчика за счет ранней и бесперебойной поставки ценного программного обеспечения;
* изменение требований приветствуется даже в конце разработки (это может повысить конкурентоспособность полученного продукта);
* частая поставка работающего программного обеспечения (каждые пару недель или пару месяцев с предпочтением меньшего периода);
* общение представителей бизнеса с разработчиками должно быть ежедневным на протяжении всего проекта;
* проекты следует строить вокруг заинтересованных людей, которых следует обеспечить нужными условиями работы, поддержкой и доверием;
* самый эффективный метод обмена информацией в команде – личная встреча;
* работающее программное обеспечение – лучший измеритель прогресса;
* спонсоры, разработчики и пользователи должны иметь возможность поддерживать постоянный темп на неопределённый срок;
* постоянное внимание к техническому совершенству и хорошему проектированию увеличивают гибкость;
* простота, как искусство не делать лишней работы, очень важна;
* лучшие требования, архитектура и проектные решения получаются у самоорганизующихся команд;
* команда регулярно обдумывает способы повышения своей эффективности и соответственно корректирует рабочий процесс.

Из минусов можно отметить отсутствие начального плана, так как он меняется на ходу. Кроме того, часто может происходить переписывание кода из-за изменчивости требований. Также разработчики стараются делать максимально просто и быстро при таком подходе, из-за чего накапливается технический долг и снижение качества продукта. Также такой код может перестать работать при дальнейшем изменении.

**Методология XP**

Экстремальное программирование (XP) – это упрощенная методология организации разработки программ для небольших и средних по размеру команд разработчиков, занимающихся созданием программного продукта в условиях неясных или быстро меняющихся требований.

Основными целями XP являются повышение доверия заказчика к программному продукту путем предоставления реальных доказательств успешности развития процесса разработки и резкое сокращение сроков разработки продукта. При этом XP сосредоточено на минимизации ошибок на ранних стадиях разработки. Это позволяет добиться максимальной скорости выпуска готового продукта и дает возможность говорить о прогнозируемости работы. Практически все приемы XP направлены на повышение качества программного продукта.

Основными принципами являются:

**Итеративность**. Разработка ведется короткими итерациями при наличии активной взаимосвязи с заказчиком. Итерации как таковые предлагается делать короткими, рекомендуемая длительность – 2-3 недели и не более 1 месяца.

**Простота решений**. Принимается первое простейшее рабочее решение. Экстремальность метода связана с высокой степенью риска решения, обусловленного поверхностностью анализа и жестким временным графиком. Реализуется минимальный набор главных функций системы на первой и каждой последующей итерации; функциональность расширяется на каждой итерации.

* **Интенсивная разработка малыми группами** (не больше 10 человек) и парное программирование (когда два программиста вместе создают код на одном общем рабочем месте), активное общение в группе и между группами. Все это нацелено на как можно более раннее обнаружение проблем (как ошибок, так и срыва сроков).
* **Обратная связь с заказчиком**, представитель которого фактически вовлечен в процесс разработки.
* Достаточная степень смелости и желание идти на риск.

##### **Приемы XP (практики)**

* **Планирование процесса**. Вся команда разработчиков собирается вместе, принимается коллективное решение о том, какие свойства системы будут реализованы в ближайшей итерации. Трудоемкость реализации каждого свойства определяется самими программистами.
* **Тесное взаимодействие с заказчиком**. Представитель заказчика должен быть членом XP-команды.
* **Общесистемные правила именования.** Хорошие системные правила именования предполагают простоту именования классов и переменных.
* **Простая архитектура**. Любое свойство системы должно быть реализовано как можно проще.
* **Рефакторинг**. Это оптимизация существующего кода с целью его упрощения, Такая работа должна вестись постоянно.
* **Коллективное владение кодом**. Каждый программист в коллективе должен иметь доступ к коду любой части системы и право вносить изменения в любой код.
* **Единые стандарты кодирования**. Стандарты кодирования нужны для обеспечения других практик: коллективного владения кодом, парного программирования и рефакторинга.Все члены команды должны договориться об общих стандартах кодирования.
* **Небольшие релизы**. Минимальная итерация – один день; максимальная – месяц. Чем чаще осуществляются релизы, тем больше недостатков системы будет выявлено.
* **Непрерывная интеграция**. Интеграция новых частей системы должна происходить как можно чаще, как минимум раз в несколько часов. Основное правило интеграции следующее: интеграцию можно производить, если все тесты проходят успешно.
* **Тестирование**. В отличие от большинства остальных методологий тестирование в XP – одно из важнейших составляющих. Экстремальный подход предполагает, что тесты пишутся до написания кода.
* Процесс XP является неформальным, но требует высокого уровня самодисциплины. Если это правило не выполняется, то XP мгновенно превращается в хаотичный и неконтролируемый процесс. XP не требует от программистов написания множества отчетов и построения массы моделей. В XP каждый программист считается квалифицированным работником, который профессионально и с большой ответственностью относится к своим обязанностям. Если в команде этого нет, то внедрять XP абсолютно бессмысленно – лучше для начала заняться перестройкой команды.

**Таблица сравнения моделей**

В таблице 1 представлены сравнения моделей и методологий ЖЦ.

Таблица 1 – Сравнения моделей и методологий ЖЦ

| Критерии | Методологии и модели | | | | |
| --- | --- | --- | --- | --- | --- |
| Scrum | Итерационная | XP | Инкрементальная | Гибкая |
| Критерий успеха | Успешность определяется по доставке ценности для клиента в каждой итерации, а также в завершенном продукте в конце проекта. | Успешность зависит от того, насколько хорошо итерации выполняются и насколько соответствуют требованиям клиента. | Успех достигается при выполнении требований клиента в каждой итерации и при использовании лучших практик программирования. | Успешность определяется по степени достижения поставленных целей в каждом этапе и насколько хорошо продукт соответствует требованиям клиента. | Успешность определяется по способности команды быстро реагировать на изменения требований клиента и доставлять ценность в каждой итерации. |
| Известность рисков, их просчет | Риски определяются на каждом этапе итерации и учитываются при планировании. | Риски учитываются на каждом этапе, но не всегда могут быть предсказаны заранее. | Риски учитываются на каждом этапе. | Риски учитываются на каждом этапе, но не всегда могут быть предсказаны заранее. | Риски учитываются на каждом этапе итерации. Кроме того, гибкая методология позволяет быстро реагировать на появление новых рисков. |
| Наличие документации | Документация минимальна и фокусируется на пользовательском опыте и требованиях. | Документация является важным компонентом итерационного процесса и должна быть поддержана на каждом этапе. | XP предусматривает создание и поддержание требований к продукту, а также кодирование и тестирование, что может потребовать некоторой документации. | Эта методология требует достаточного количества документации, так как каждый прототип может иметь свои требования и тестовые планы. | В Agile создаются документы, которые необходимы для обеспечения прозрачности и обмена информацией между членами команды. |
| Вовлеченность заказчика | Заказчик является членом команды и принимает активное участие в планировании и обсуждении продукта. Кроме того, он участвует в регулярных встречах, на которых обсуждаются промежуточные результаты. | Заказчик обычно не принимает непосредственного участия в разработке, но регулярно получает отчеты о ходе проекта и дает обратную связь. | Заказчик также является членом команды и участвует в ежедневных встречах, на которых обсуждаются проблемы и принимаются решения. Он также участвует в оценке и тестировании продукта. | Заказчик принимает активное участие в планировании и тестировании каждой новой итерации, но не участвует в каждом этапе разработки. | Заказчик является одним из ключевых участников команды и участвует в планировании и обсуждении продукта наравне с разработчиками. Он также дает обратную связь и участвует в оценке и тестировании продукта. |
| Известны ли требования, как изменяются | Требования изначально не полностью известны, могут меняться | Требования будут определены на стадии планирования проекта и затем разработка будет осуществляться по итерациям, в которых требования могут быть уточнены и изменены. | Требования известны, но могут изменяться. | Требования должны быть определены на первой стадии и могут меняться на следующих этапах. | Изменчивые требования и постоянное взаимодействие с заказчиком. |
| Вовлечение команды, много ли надо сотрудников, нужны ли постоянно новые | Активно вовлекает команду в процесс разработки и предоставляет ей большую свободу в выборе способов выполнения задач. Количество сотрудников в команде обычно составляет от 5 до 9 человек. Новые сотрудники могут добавляться к команде только в конце спринта. | Не определяет жесткие правила для команды, но требует дополнительных усилий со стороны команды для оценки проекта и определения ролей и ответственности в проекте. Количество сотрудников обычно составляет от 3 до 10 человек, в зависимости от размера проекта. | XP активно вовлекает команду в процесс разработки и предоставляет ей большую свободу в выборе способов выполнения задач. Количество сотрудников в команде обычно составляет от 5 до 12 человек. | Предполагает, что команда состоит из 5-10 сотрудников, и количество сотрудников может меняться в зависимости от размера проекта и его потребностей. Новые сотрудники могут быть добавлены в любой момент разработки. | Активно вовлекает команду в процесс разработки и предоставляет ей большую свободу в выборе способов выполнения задач. Количество сотрудников может колебаться в зависимости от размера проекта и его потребностей. Новые сотрудники могут быть добавлены в любой момент разработки. |
| Расширение проекта, сколько стоит денег и времени | Методология Scrum позволяет гибко вносить изменения в проект, однако, это может затруднить планирование бюджета и времени. Оценка стоимости новых изменений может быть сложной из-за динамической природы Scrum. | Гибкая в отношении изменений в проекте, но каждый этап должен быть завершен, прежде чем начнется следующий. Это может затруднить внесение изменений в середине проекта. | Разработана для быстрой и гибкой разработки, что позволяет вносить изменения в проект. Однако, каждое изменение должно проходить тестирование и может привести к увеличению стоимости проекта. | Предполагает поэтапное добавление функциональности, что облегчает внесение изменений в проект. Однако, каждая инкрементальная итерация может привести к увеличению стоимости проекта. | Предоставляет гибкость в изменении требований и функциональности проекта. Однако, это может привести к увеличению стоимости проекта. |
| Решает ли скорость или качество | Ориентирована на быстрое создание рабочего продукта высокого качества. Скорость и качество считаются равноправными приоритетами, но при этом качество обычно не страдает от скорости. | Приоритет уделяется скорости выполнения задач, поскольку каждая итерация должна заканчиваться работоспособным продуктом. Однако, качество также важно, и проверки и балансирование влияния скорости на качество осуществляются на каждой итерации. | Качество продукта считается самым важным критерием успеха, а скорость является неотъемлемой частью процесса. Следование практикам XP обеспечивает высокое качество продукта, но может замедлять процесс разработки. | Ориентирована на создание рабочего продукта в быстром темпе. Скорость выполнения задач имеет приоритет перед качеством. Качество продукта, как правило, улучшается по мере того, как проект развивается, и происходит проверка на каждой стадии. | В гибких методологиях скорость и качество являются равноправными и взаимодополняющими критериями успеха. Важно выполнять работу быстро, но при этом обеспечивать высокое качество продукта. Однако, различные методы гибкой разработки могут уделять больший приоритет одному из критериев в зависимости от контекста проекта. |

**Выбор подходящей модели**

В таблице 1 было выполнено сравнение 5 моделей по различным критериям.

Для создания интернет-магазина была выбрана гибкая методология Agile. Это связано с тем, что в создании интернет-магазина могут возникать различные изменения требований и улучшения, которые можно легко внести с помощью гибкой методологии. Кроме того, гибкая методология позволяет быстро выпускать новые версии продукта и обеспечивать высокое качество разработки.

Важным аспектом при создании интернет-магазина является также вовлеченность заказчика. Гибкая методология позволяет заказчику активно участвовать в процессе разработки, выражать свои требования и получать быстрые результаты.

Кроме того, для создания интернет-магазина может потребоваться большое количество сотрудников, так как нужно заниматься не только программированием, но и дизайном, контентом, маркетингом и т.д. Гибкая методология позволяет эффективно управлять командой разработчиков и других сотрудников, снижать риски и сокращать время разработки.

Наконец, гибкая методология также позволяет эффективно решать вопросы качества и скорости разработки, поскольку ее основная задача –обеспечивать быстрое и гибкое реагирование на изменения и улучшения.

**Выводы**

В результате выполнения данного индивидуального домашнего задания задания было представлено описание пяти выбранных моделей ЖЦ ПО.

После выполнения сравнительного анализа наиболее подходящей методологией для разработки сайта интернет-магазина стала методология AGILE.

Это связано с тем, что проект разрабатывается небольшой командой из 3-4 студентов с минимальным опытом работы. Также за счет неопытности нельзя требовать скорости разработки, но можно требовать наличие рабочего продукта на каждой стадии разработки с использованием простых, но рабочих решений.