УДК 004.413

АНАЛИЗ МОДЕЛЕЙ ЖИЗНЕННОГО ЦИКЛА ДЛЯ КРОССПЛАТФОРМЕННОЙ РАЗРАБОТКИ КОРПОРАТИВНОГО ИНФОРМАЦИОННОГО ПОРТАЛА

Пименов А.А., Колесников Л.А., Панов А.В.

МИРЭА – Российский технологический университет, 119454, Россия, г. Москва, проспект Вернадского, 78, e-mail: aa.pimenov@yandex.ru, nickname.text@yandex.ru, actual1243@yandex.ru

При разработке кроссплатформенных систем, таких как корпоративный информационный портал важным является правильный выбор модели жизненного цикла. Для его выбора необходимо сформировать список требований, отвечающий условиям конкретной разрабатываемой системы. В данной статье будет сформирован список этих требований, а также рассмотрены и проанализированы популярные модели жизненного цикла для понимания плюсов и минусов каждой, опираясь на требования.

Ключевые слова: жизненный цикл, стартап, каскадная модель, инкрементная модель, спиральная модель, корпоративный информационный портал, ценности agile, scrum.

ANALYSIS OF LIFE CYCLE MODELS FOR CROSS-PLATFORM DEVELOPMENT OF CORPORATE INFORMATION PORTAL

Pimenov A.A., Kolesnikov L.A., Panov A.V.

MIREA – Russian Technological University, 119454, Moscow, 78 Vernadskogo Avenue, Russia e-mail: aa.pimenov@yandex.ru, nickname.text@yandex.ru, actual1243@yandex.ru

When developing cross-platform systems, such as a corporate information portal, it is important to choose the right lifecycle model. To select it, you need to create a list of requirements that meet the conditions of the specific system being developed. This article will create a list of these requirements, as well as review and analyze popular life cycle models to understand the pros and cons of each, based on the requirements.

Key words: lifecycle, startup, cascade model, incremental model, spiral model, corporate information portal, agile values, scrum.

Введение

Любое программное обеспечение, портал или самостоятельный продукт имеют конкретные стадии развития, называемые фазами жизненного цикла. Точное представление этих фаз дает возможность эффективно ставить задачи для достижения целей проекта. Перед началом любых работ над проектом необходимо выбрать модель его жизненного цикла, согласно идеям и целям конечного продукта.

В свою очередь, жизненный цикл проекта (от англ. Project Life Cycle) представляет собой период времени между возникновением идеи проекта до его полной реализации [2]. При этом обозначение рамок начала и конца жизненного цикла остается за участниками проекта.

Требования к модели жизненного цикла

При кроссплатформенной разработке корпоративного информационного портала в небольших командах или стартапах, где команда разработки также выступает в роли заказчика, возникает вопрос о выборе наиболее подходящей модели жизненного цикла, согласно нижеперечисленным требованиям:

– Быстрый выпуск. Это необходимо для скорейшего выхода продукта на рынок и знакомства потребителя с ним. Благодаря первым отзывам можно понять ценность продукта для конечного пользователя.

– Возможность вносить коррективы на протяжении всего процесса разработки. На основе первоначальной оценки продукта пользователя можно произвести дополнительный анализ к требованиям и в случае необходимости внести коррективы.

– Взаимодействие между командами разных платформ. Во время кроссплатформенной разработки важной частью является коммуникация между разными командами для помощи в преодолении возникающих

трудностей. Это необходимо для одновременной разработки функциональной составляющей для разных платформ.

– Постоянный анализ технического прогресса и новых тенденций. Важно поддерживать актуальность технологических решений для улучшения конкурентоспособности продукта.

– Небольшие затраты. Стартапы или малые команды зачастую ограничены в бюджете.

– Простота внедрения. Небольшие проекты ограничены во временных рамках.

– Простота использования. Следование определенной модели жизненного цикла не должно отнимать много времени от реализации проекта.

В этой статье будут рассмотрены и проанализированы популярные модели жизненных циклов программного обеспечения и будет выбрана одна модель для дальнейшего использования при создании корпоративного информационного портала.

Каскадная модель жизненного цикла программного обеспечения

Чаще всего именно такую модель называют классическим жизненным циклом. Кроме этого, ее часто называют водопад (от английского waterfall). Главная идея данной модели заключается в том, что процесс разработки выглядит как непрерывный поток, проходящий последовательно по каждой из заранее определенных фаз. Новая фаза начинается только после полного завершения работ на предыдущей, а возврат на прошедшие фазы недопустим. Впервые ее концепцию описал Уинстон Уокер Ройс в 1970 году в своей статье. Со временем каскадная модель была регламентирована различными нормативными документами во всем мире. Например, в России стандартами серии ГОСТ 34 [1]. Каскадная модель включает в себя следующие этапы:

– Формирование требований.

– Анализ требований, спецификация.

– Проектирование.

– Реализация (кодирование).

– Тестирование и отладка.

– Сопровождение (внедрение и отладка).

Этапы могут как сливаться, так и дробиться на более точечные.

Данной модели присуща точная формулировка требований на этапе их формирования и согласования фиксируется в ТЗ без возможности последующих корректировок. После завершения каждого этапа формируется документация. Этапы проходят последовательно без возможности возврата к предыдущим. Такая модель может подойти при разработке высокоточного программного обеспечения с большим набором задач, в котором все требования возможно точно определить и согласовать заранее [3].

Каскадная модель имеет ряд преимуществ:

– Простота и ясность во время внедрения и использования.

– Последовательное выполнение всех этапов позволит рассчитать бюджет и запланировать сроки.

Кроме этого, данная модель имеет и ряд недостатков:

– Потенциальные и фактические ошибки и проблемы обнаруживаются достаточно поздно, лишь на этапе тестирования.

– Отсутствие связи между этапами.

– Определение требований в начале проекта не позволит их менять во время разработки.

– Конечный продукт невозможно разбить на части, и он предоставляется заказчику только после завершения проекта.

Большое количество документации, которая может оказаться избыточной.

Зачастую, реальная разработка подвергается отклонению от стандартной последовательности шагов, а требования заказчика могут меняться по мере разработки системы не только из-за желания заказчика, но и из-за изменений во внешней среде.

Инкрементная модель жизненного цикла программного обеспечения

Данная модель объединяет в себе этапы последовательной каскадной модели и итерационной стратегии разработки. Главная идея заключается в том, что разработка делится на несколько инкрементов или версий продукта. В начале, в первой версии разрабатывается минимальный жизнеспособный продукт (от английского Minimal Value Product). Он обладает минимальным набором функциональных возможностей, удовлетворяющих заказчика. С каждой версией продукт модифицируется, он обретает дополнительный функционал, вид и прочее. Версии передаются заказчику по мере готовности, в свою очередь, это позволяет начать использовать продукт с минимальными временными затратами. Данная модель позволяет постепенно приобщить пользователя к новому продукту. Весь этот процесс завершается тогда, когда будет создана полная система [5].

Все начинается с формирований требований и их анализа, затем каждый инкремент включает в себя следующие этапы:

– Проектирование.

– Реализация (кодирование).

– Тестирование и отладка.

– Выпуск.

На рис. 1 показано графическое представление данной модели жизненного цикла.

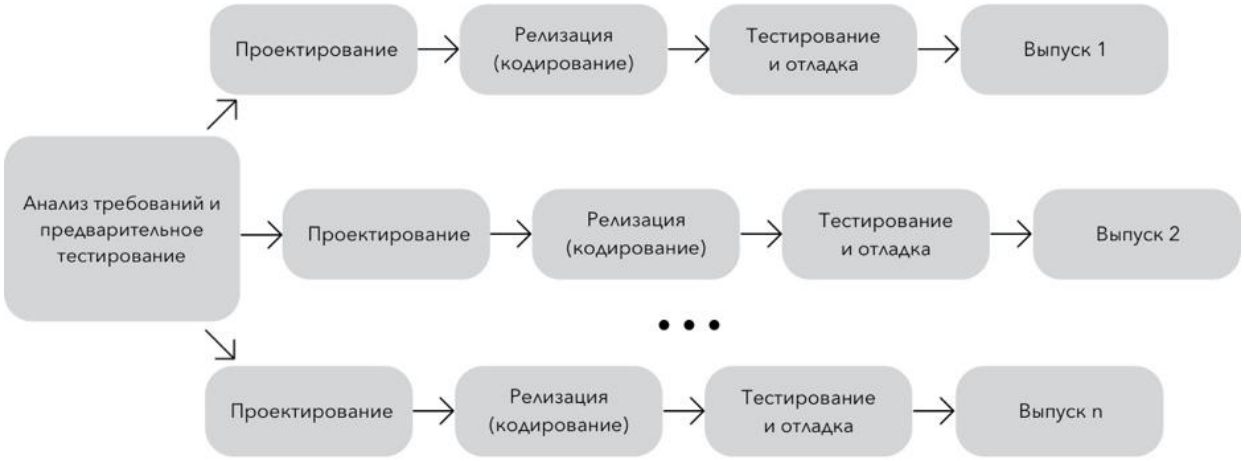

Рис. 1. Инкрементная модель жизненного цикла

Данную модель используют, когда все основные требования к системе определены, но некоторые из них не требуют первоочередной разработки либо необходимо быстро вывести продукт. Инкрементная модель имеет ряд преимуществ:

– На ранних стадиях заказчик получает работающий продукт.

– Небольшие версии продукта быстро тестируются и сразу можно внести правки.

– Простота в использовании так как все итерации определены заранее.

Кроме этого, данная модель имеет и ряд недостатков:

– Итоговая стоимость разработки может увеличиться.

– Разбиение проекта на инкременты требует больших временных затрат на начальных этапах.

– При добавлении дополнительного функционала могут возникнуть проблемы с архитектурой.

– Требования к продукту необходимо формулировать на начальном этапе для разбиения проекта на инкременты.

Чаще всего такая модель применятся при разработке больших веб-приложений. При разработке рискованного продукта можно на первой итерации понять необходимость в нем.

Спиральная модель жизненного цикла программного обеспечения

Данная модель была предложена Барри Боэмом в 1986 году. Модель похоже на инкрементную, но в отличие от инкрементной здесь уделяется существенное внимание на анализ рисков проекта. Модель содержит отдельную стадию для оценки рисков. Графически такая модель представляет собой плоскость, которая делится на 4 области, каждая из которых представляет отдельные этапы разработки ПО:

– Формирование задач.

– Обозначение рисков и способов борьбы с ними.

– Кодирование и тестирование.

– Планирование следующих итераций [6].

На рис. 2 показано графическое представление данной модели жизненного цикла.

К преимуществам спиральной модели можно отнести следующее:

– Гибкость, которая позволяет дополнять и перерабатывать функционал.

– Быстрое создание рабочего прототипа.

– Возможность корректировки используемых технологий на каждом витке.

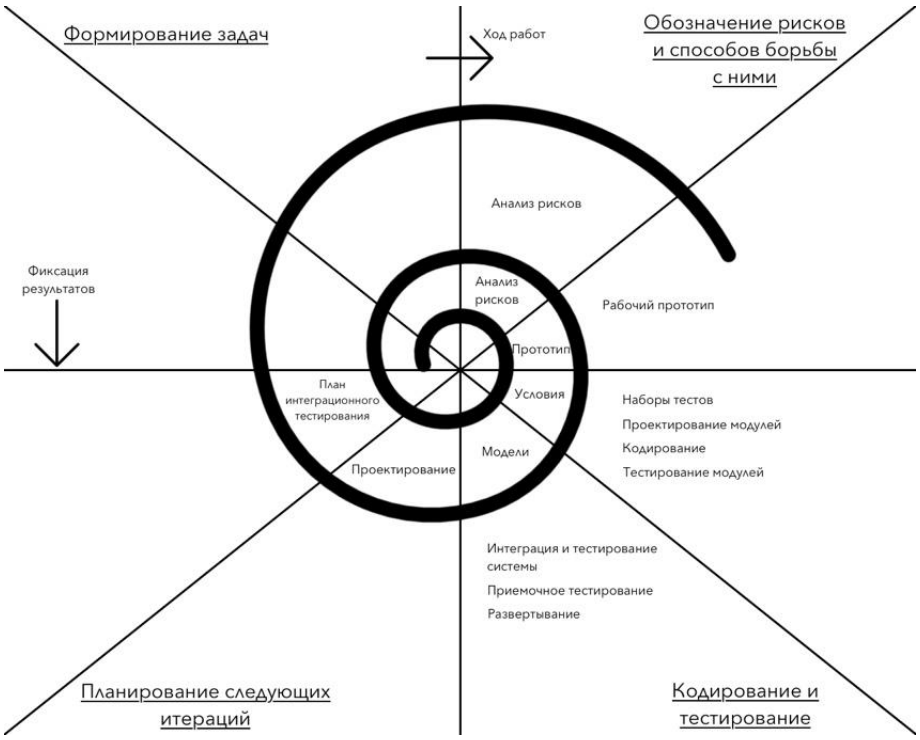

Рис. 2. Спиральная модель жизненного цикла

Но также у модели есть недостатки:

– Дорогая в использовании. Например, для регулярного управления рисками необходим специалист соответствующего уровня и квалификации.

– Наступление следующего этапа может быть долгим за счет сложного определения критериев перехода.

– Сложная структура модели может привести к затруднениям ее использования разработчиками.

Эта модель позволяет вести гибкое проектирование, а также добавлять дополнительный функционал на более поздних этапах. Вероятность ошибиться при разработке снижается за счет отдельного этапа по анализу рисков. Но такой подход является ресурсно-затратным и не подойдет для стартапов и небольших компаний.

Scrum модель жизненного цикла программного обеспечения

Scrum является одним из возможных способов использования Agile. Agile является гибкой методологией разработки. Она включает в себя совокупность практик, которые основаны на 12 принципах манифеста гибкой разработки программного обеспечения. Именно в феврале 2001 года в штате Юта США был представлен данный манифест. Известно то, что до этого события многие компании уже использовали гибкие методологии разработки, но популяризация их произошла после принятия манифеста.

Стоит отметить, что Agile не является единственным подходом к разработке программного обеспечения, а представляет собой ценности и практики, которые разработчики используют в своих проектах и в организации деятельности.

Всего в Agile 4 ценности:

– Люди и их взаимодействие важнее процессов и инструментов.

– Работающий продукт важнее исчерпывающей документации.

– Сотрудничество с заказчиком важнее согласований условий контракта.

– Готовность к изменениям важнее, чем следование плану [4].

Применение ценностей позволит увеличить скорость разработки.

Одной из отличительных особенностей Scrum, является итеративность. Сам процесс разработки состоит из спринтов, по результатам каждого из которых получается рабочий продукт. Жизненный цикл Scrum можно представить с помощью 5 этапов:

– Формирование бэклога продукта. На данном этапе формируется список требований, а также происходит оценка этих требований и сортировка их по важности.

– Планирование спринта и формирование бэклога спринта. В это время решается то, как долго будет длиться текущий спринт и какие требования из бэклога он покроет.

– Спринт. На этапе самого спринта происходит непосредственно процесс разработки.

– Тестирование и представление продукта. Тестирование является важным составляющим Scrum и ему уделяется особое внимание, так как по окончанию спринта должен получиться рабочий продукт.

– Ретроспектива. Данный этап позволяет проанализировать отзывы о продукте, которые были получены на предыдущем этапе. На основе отзывов бэклог продукта может возрасти. Также этот этап служит для планирования следующих спринтов [7].

На рис. 3 показано графическое представление данной модели жизненного цикла.

Рис. 3. Scrum модель жизненного цикла

Таким образом, Scrum является отличным инструментом для внедрения гибкой методологии разработки. Это поможет в дальнейшем продукту оперативно изменяться и дорабатываться, если это будет необходимо. Так как по завершению каждого спринта получается рабочий продукт, который уже содержит функционал, то на этапе ретроспективы всегда можно оценить достигнуты ли цели и что можно исправить или доработать. Внедрение Scrum лучше всего производить на этапе выбора модели жизненного цикла, так как переход с одной модели на другую может быть очень сложным и затратным.

Сравнение моделей жизненного цикла.

На основе анализа представленных моделей жизненного цикла и согласно требованиям, сформированным в начале, была построена таблица 1 в которой отражены плюсы и минусы каждой из моделей.

Таблица 1. Сравнение моделей жизненного цикла

Требования/Модели	Каскадная	Инкрементная	Спиральная	Scrum
Быстрый выпуск	–	+	+	+
Дополнительные коррективы	–	–	+	+
Взаимодействие команд	–	+	–	+
Анализ технического прогресса	–	–	+	+
Небольшие затраты	+	–	–	–
Простота внедрения	+	–	–	+
Простота использования	+	+	–	+

Согласно полученным данным можно сделать вывод, что наиболее подходящая модель для кроссплатформенной разработки корпоративного информационного портала в небольших командах является Scrum, которая основана на Agile ценностях. Scrum модель жизненного цикла сложна для внедрения в больших компания, за счет привлечения отдельных высококвалифицированных специалистов, но в рамках небольших команд или проектов они не требуются. Таким образом команды могут эффективно построить свой рабочий процесс с использованием инструментов Scrum.

Заключение

В данной статье были рассмотрены и проанализированы 4 наиболее популярные модели жизненного цикла программного обеспечения, кроме этого, были сформированы требования, отвечающие условиям разработки кроссплатформенного корпоративного информационного портала в небольших командах или стартапах. На основе этого анализа, был сделан вывод, что будут эффективны Agile ценности, которые отражены в модели жизненного цикла Scrum.

Список литературы

1. ГОСТ 34.601–90. Информационная технология. Комплекс стандартов на автоматизи- рованные системы. Автоматизированные системы. Стадии создания. М. : Госстандарт России, 2009. - 6 с.

2. Добряк П.В., Ульянова Е.А., Берг Д.Б. Модели жизненного цикла. Учебное пособие / - Уральский федеральный университет, 2014. – 120 с.

3. Кумагина Е.А. Модели жизненного цикла и технологии проектирования программного обеспечения / ННГУ, 2017. – 40 с.

4. Мартин Р. Чистый Agile. Основы гибкости / -СПб.: Питер, 2020. – 272 с.

5. Орлов С.А. Технологии разработки программного обеспечения: Учебник / – СПб.: Питер, 2002. – 464 с.

6. Самочкин В. Н. Фазы жизненного цикла изделий и планирование гибкого развития предприятия / В. Н. Самочкин / Маркетинг в России и за рубежом. 1998. - 15 с.

7. Швабер К. Гибкое управление продуктом и бизнесом / - Альпина Паблишер, 2019. – 240 с.

References

1. GOST 34.601–90. Information technology. Set of standards for automated systems. Automated systems. Stages of creation. M.: Gosstandart of Russia, 2009 .- 6 p.

2. Dobryak P.V., Ulyanova E.A., Berg D.B. Life cycle models. Textbook / - Ural Federal University, 2014 .- 120p.

3. Kumagina E.A. Life cycle models and software design technologies / NNSU, 2017 .- 40 p.

4. Martin R. Pure Agile. Fundamentals of Flexibility / -SPb .: Peter, 2020 .- 272 p.

5. Orlov S.A. Software development technologies: Textbook / - SPb .: Peter, 2002. - 464 p.

6. Samochkin V.N. Phases of the life cycle of products and planning of flexible development of the enterprise / V.N. Samochkin / Marketing in Russia and abroad. 1998. – 15 p

7. Shvaber K. Flexible product and business management / - Alpina Publisher, 2019. - 240 p.