БЕЛОРУССКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ

ЭКОНОМИЧЕСКИЙ ФАКУЛЬТЕТ

Кафедра экономической информатики

БОУРОШ Андрей Игоревич

СТРАТЕГИИ ПРОДУКТОВОЙ РАЗРАБОТКИ МОБИЛЬНЫХ ПРИЛОЖЕНИЙ НА ОСНОВЕ ИСПОЛЬЗОВАНИЯ ГИБКИХ МЕТОДОЛОГИЙ AGILE

Магистерская диссертация

специальность 1-25 81 10 «Экономическая информатика»

Научный руководитель Марушко Дмитрий Александрович
кандидат экономических наук,
доцент

Допущена к защите
«___» _____ 2019 г.
Зав. кафедрой экономической информатики
_____Марушко Д.А.
кандидат экономических наук, доцент

Минск, 2019

ОГЛАВЛЕНИЕ

ОБЩАЯ ХАРАКТЕРИСТИКА РАБОТЫ

Ключевые слова: УПРАВЛЕНИЕ ПРОЕКТАМИ, РАЗРАБОТКА ПРОГРАММНОГО ОБЕСПЕЧЕНИЯ, МОБИЛЬНАЯ РАЗРАБОТКА, МЕТОДОЛОГИЯ, УПРАВЛЕНИЕ БИЗНЕС ПРОЦЕССАМИ, ГИБКИЕ МЕТОДОЛОГИИ, МОТИВАЦИЯ, АВТОМАТИЗАЦИЯ, КАСКАДНАЯ МОДЕЛЬ, ИТЕРАТИВНО-ИНКРЕМЕНТНАЯ МОДЕЛЬ, СКРАМ, ПРОДУКТОВАЯ СТРАТЕГИЯ.

Цель исследования: исследовать подходы к управлению проектами и проблему выбора методологии разработки для проектов по разработке мобильных приложений, разработать предложения по выбору методологии управления проектом, рассмотреть проблемы повышения эффективности управления проектами и предложить подходы к формированию продуктовой стратегии разработки мобильных приложений.

Объект исследования: продуктовая разработка мобильных приложений.

Предмет исследования: методологии управления проектами, бизнес процессы управления проектами и инструментарий для их организации и оптимизации.

Полученные результаты и их новизна: сравнительный анализ методологий управления проектами. Рекомендации по выбору методологий управления проектами и их внедрению с использованием программно-технического инструментария управления проектами. Сформулированы подходы к формированию стратегии продуктовой разработки мобильных приложений.

Область возможного практического применения: использование полученных результатов для организации и совершенствования процессов управления проектами по разработке мобильных приложений. Использование рекомендаций для разработки стратегии продуктовой разработки. Использование в качестве пособия для изучения методологий и организации рабочих групп в учебных заведениях.

Структура магистерской работы: 68 страниц, 4 таблицы, 8 рисунков, количество использованных библиографических источников 32.

АГУЛЬНАЯ ХАРАКТАРЫСТЫКА ПРАЦЫ

Ключавыя словы: КІРАВАННЕ ПРАЕКТАМІ, РАСПРАЦОЎКА ПРАГРАМНАГА ЗАБЕСПЯЧЭННЯ, МАБІЛЬНАЯ РАСПРАЦОЎКА, МЕТАДАЛОГІЯ, КІРАВАННЕ БІЗНЕС ПРАЦЭСАЎ, ГНУТКАЯ МЕТАДАЛОГІЯ, МАТЫВАЦЫЯ, АЎТАМАТЫЗАЦЫЯ, КАСКАДНАЯ МАДЭЛЬ, ІТЭРАТЫЎНА-ІНКРЭМЕНТНАЯ МАДЭЛЬ, СКРАМ, ПРАДУКТОВАЯ СТРАТЭГІЯ.

Мэта даследавання: даследаваць падыходы да кіравання праектамі і праблему выбару метадалогіі распрацоўкі для праектаў па распрацоўцы мабільных прыкладанняў, распрацаваць прапановы па выбару метадалогіі кіравання праектам, разглядзець праблемы павышэння эфектыўнасці кіравання праектамі і прапанаваць падыходы да фарміравання прадуктовай стратэгіі распрацоўцы мабільных прыкладанняў.

Аб'ект даследавання: прадуктовая распрацоўка мабільных прыкладанняў.

Прадмет даследавання: метадалогіі кіравання праектамі, бізнэс-працэсы кіравання праектамі і інструменты для іх арганізацыі і аптымізацыі.

Атрыманыя вынікі і іх навізна: параўнальны аналіз метадалогій кіравання праектамі. Рэкамендацыі па выбару метадалогій кіравання праектамі і іх ўкараненню разам з выкарыстаннем праграмна-тэхнічных інструментаў кіравання праектамі. Сфармуляваны накірункі да фарміравання стратэгіі прадуктовай распрацоўцы мабільных прыкладанняў.

Вобласць магчымага практычнага прымянення: выкарыстанне атрыманых вынікаў для арганізацыі і ўдасканалення працэсаў кіравання праектамі па распрацоўцы мабільных прыкладанняў. Выкарыстанне рэкамендацый для распрацоўкі стратэгіі прадуктовай распрацоўцы. Выкарыстанне ў якасці дапаможніка для вывучэння метадалогій і арганізацыі груп у навучальных установах.

Структура магістарскай працы: 68 старонак, 4 табліцы, 8 малюнкаў, колькасць выкарыстаных бібліяграфічных крыніц 32.

GENERAL DESCRIPTION OF WORK

Keywords: PROJECT MANAGEMENT, SOFTWARE DEVELOPMENT, MOBILE DEVELOPMENT, METHODOLOGY, BUSINESS PROCESS MANAGEMENT, AGILE METHODOLOGIES, MOTIVATION, AUTOMATION, WATERFALL, ITERATIVE AND INCREMENTAL DEVELOPMENT, SCRUM, PRODUCT STRATEGY.

The purpose of the research: to explore approaches to project management and the problem of choosing the methodology of mobile application development projects, to develop proposals for the choice of project management methodology, to consider the problems of improving the efficiency of project management and suggest approaches to the formation of product strategy for mobile application development.

Object of research: product development of mobile applications.

The subject of the research: project management methodologies, business processes of project management, collaboration tools for organization and optimization of project management.

The results obtained and their novelty: comparative analysis of project management methodologies. Recommendations for project management methodologies selection and their implementation with use of software tools for project management. Formulated approaches to the formation of product mobile development strategies.

Area of possible practical application: use of the results for creating and improving of project management processes for the development of mobile applications. Use recommendations to develop a product management strategy. Use as a guide for studying methodologies and organizing working groups in educational institutions.

Master's work structure: 68 pages, 4 tables, 8 figures, number of bibliographic sources used 32.

РЕЗУЛЬТАТЫ ТЕСТИРОВАНИЯ МАГИСТЕРСКОЙ ДИССЕРТАЦИИ НА ПЛАГИАТ (РАСПЕЧАТКА С САЙТА WWW.ANTIPLAGIAT.RU)

АНТИПЛАГИАТ
ТВОРИТЕ СОБСТВЕННЫМ УМОМ

Отчет о проверке на заимствования №1

Автор: Bourosh Andrew bourosh@gmail.com / ID: 6824431
Проверяющий: Bourosh Andrew (bourosh@gmail.com / ID: 6824431)

Отчет предоставлен сервисом «Антиплагиат»- http://users.antiplagiat.ru

ИНФОРМАЦИЯ О ДОКУМЕНТЕ

№ документа: 1
Начало загрузки: 04.06.2019 02:20:53
Длительность загрузки: 00:00:04
Имя исходного файла: agile-curated
Размер текста: 1502 кБ
Символов в тексте: 99180
Слов в тексте: 12191
Число предложений: 573

ИНФОРМАЦИЯ ОБ ОТЧЕТЕ

Последний готовый отчет (ред.)
Начало проверки: 04.06.2019 02:20:57
Длительность проверки: 00:00:03
Комментарии: не указано
Модули поиска: Модуль поиска Интернет

ЗАИМСТВОВАНИЯ	ЦИТИРОВАНИЯ	ОРИГИНАЛЬНОСТЬ
11,14%	0%	88,86%

Заимствования — доля всех найденных текстовых пересечений, за исключением тех, которые система отнесла к цитированиям, по отношению к общему объему документа.
Цитирования — доля текстовых пересечений, которые не являются авторскими, но система посчитала их использование корректным, по отношению к общему объему документа. Сюда относятся оформленные по ГОСТу цитаты; общеупотребительные выражения; фрагменты текста, найденные в источниках из коллекций нормативно-правовой документации.
Текстовое пересечение — фрагмент текста проверяемого документа, совпадающий или почти совпадающий с фрагментом текста источника.
Источник — документ, проиндексированный в системе и содержащийся в модуле поиска, по которому проводится проверка.
Оригинальность — доля фрагментов текста проверяемого документа, не обнаруженных ни в одном источнике, по которым шла проверка, по отношению к общему объему документа.
Заимствования, цитирования и оригинальность являются отдельными показателями и в сумме дают 100%, что соответствует всему тексту проверяемого документа.
Обращаем Ваше внимание, что система находит текстовые пересечения проверяемого документа с проиндексированными в системе текстовыми источниками. При этом система является вспомогательным инструментом, определение корректности и правомерности заимствований или цитирований, а также авторства текстовых фрагментов проверяемого документа остается в компетенции проверяющего.

№	Доля в отчете	Источник	Ссылка	Актуален на	Модуль поиска
[01]	3,59%	А.Н. Зубец. Проблема бедности в современном российском обществе // ...	http://fa.ru	10 Ноя 2016	Модуль поиска Интернет
[02]	0%	А.Н. Зубец. Проблема бедности в современном российском обществе // ...	http://fa.ru	27 Окт 2016	Модуль поиска Интернет
[03]	1,49%	Просмотр ресурса	http://study.urfu.ru	05 Дек 2017	Модуль поиска Интернет

Еще источников: 13
Еще заимствований: 6,07%

ВВЕДЕНИЕ

В наши дни компании ведущие деятельность в сфере мобильной разработки подвержены постоянным и стремительным изменениям. Открытость информации, глобализация, а также невозможность защиты своих разработок от копирования, создают условия для высокой конкуренции и необходимости быстрой адаптации к изменяющимся условиям. Требования рынка задают необходимость быстро изменять способы управления и организации своих процессов, для того чтобы сохранять конкурентное преимущество. Снижение расходов на разработку продуктов и сокращение сроков - задача стоящая как перед стартапами, так и перед уже занявшими свою нишу игроками. Для того чтобы успешно функционировать в сложившихся условиях необходима хорошая организация бизнес процессов и как часть этих процессов - способы управления продуктами и проектами. Моя работа будет посвящена изучению и анализу методов управления проектами.

Нельзя оставлять в стороне и социальную сторону вопроса. Во многих странах идет процесс сокращения рабочей недели. Снижение стресса и повышение мотивации довольно непростая проблема стоящая перед компаниями, поскольку рынок труда с появлением таких сервисов как Glassdoor стал очень прозрачным и компании не уделяющие этому внимания в перспективе могут столкнуться с трудностью найма хороших специалистов, а как результат снижение качества продукта и увеличение фонда оплаты труда. Таким образом, задача выбора и разработки методов управления усложняется, так как приходится учитывать не только технические факторы, которые можно оценить математически, но и психологические, а это уже трудно поддается измерению, однако не может быть проигнорировано. Задача обеспечения психологического комфорта сотрудников может решаться разными способами, либо их комбинацией. Самым простым подходом может быть вынесение этого вопроса за рамки непосредственно производственного процесса, например различные бонусы, страховки, комфортные рабочие места и офисы. Другой подход - такая организация рабочего процесса, которая позволяет команде получать удовольствие от самого процесса работы. То есть в долгосрочной перспективе, идеальной ситуацией для компании, с точки зрения сохранения производительности

команд, является нахождение некоторого баланса бизнес целей и мотивирующих драйверов. Бизнес цели в нашем контексте это например реализация продукта, либо его части, достижение метрик. Мотивирующими драйверами для команд в этом контексте могут быть завершение запланированного объема работ, выполнение задач важных с точки зрения членов команды (они могут совпадать с бизнес целями или нет), получение обратной связи от пользователей или возможность повлиять на вектор развития проекта. Этот подход сложно реализовать при использовании классической водопадной модели, однако существуют методологии управления проектами, которые позволяют учесть это в организации бизнес процессов и развитие этих методологий является естественным результатом решения этой задачи.

Одним из самых распространенных и подходящих подходов для управления проектами в непредсказуемых условиях является Agile модель объединяющая семейство гибких методологий, особенно это касается индустрии разработки программного обеспечения. Причинами тому является такие характеристики как гибкость, отзывчивость, которые позволяют успешно справляться с внезапными, частыми, зачастую кардинальными изменениями требований к продукту [1].

В этой работе значительное внимание будет уделено исследованию существующих подходов к организации процесса разработки программного обеспечения и изучению их эволюции. Гибкие методологии разработки, на данный момент являются самыми популярными и для некоторых сфер (как например мобильная разработка) являются практически стандартом. И хотя эти подходы принято считать новыми и современными, их корни уходят в глубокое прошлое.

Другой задачей будет сравнение эффективности использования разных методологий для реализации проекта разработки мобильного приложения, а также изучение условий при которых выбор той или иной методологии является более оптимальным.

ГЛАВА 1
ТЕОРЕТИЧЕСКИЕ ОСНОВЫ УПРАВЛЕНИЯ ПРОЕКТАМИ ПО ПРОДУКТОВОЙ РАЗРАБОТКЕ МОБИЛЬНЫХ ПРИЛОЖЕНИЙ

1.1. Понятие, сущность, роль и значение управления проектами в продуктовой разработке мобильных приложений

Управление проектами в разработке программного обеспечения и мобильных приложений в частности - это особый вид управления проектами, в рамках которого происходит планирование, отслеживание и контроль за проектами по разработке программного обеспечения. Ключевым моментом в управлении проектом по разработке программного обеспечения является правильный выбор метода разработки. Особенностями данного вида управления проектами является то, что:

- конечный результат проекта по разработке программного обеспечения нематериален;
- недостаточность накопленного в данной области опыта;
- быстрое изменение используемых в проекте технологий;
- опыт управления проектами по разработке программного обеспечения часто не может быть применен к другим проектам.

Быстрое увеличение мощностей компьютеров в 60-е и 70-е годы XX века и проблемы, которые могли быть решены с их помощью, становились все сложнее. Поэтому требовались более масштабные проекты, включавшие в себя координацию труда большего числа людей и написание гораздо большего объема кода. Однако методы, применявшиеся к управлению такими проектами, были рассчитаны на решение задач в рамках намного меньших проектов. Отсутствие необходимой методологии привело к большому числу проектов, которые либо не были реализованы, либо были реализованы со значительным превышением их сроков и стоимости. Эволюция подходов к управлению проектов привела к созданию новых

моделей и методологий управления проектов, которые больше внимания акцентировали на соответствие конечного программного продукта требованиям заказчика.

Исследования неудачных проектов, показали, что наиболее распространёнными причинами были [16]:

- невыполнимые или неясно сформулированные цели проекта;
- ошибочный подсчет необходимых ресурсов;
- некорректно определенные системные требования;
- неосведомленность управляющего проектом о точном состоянии проекта;
- неуправляемые риски;
- слабое взаимодействие между заказчиком, разработчиком и пользователем;
- использование слишком новых, нестабильных технологий;
- неспособность справиться со сложностями проекта;
- слабое управление проектом;
- финансовые ограничения.

Усилия по разработке методологий управления проектами направлены на борьбу с этими причинами и создание таких систем, процессов и подходов, при которых эти трудности могут быть разрешены максимально эффективным способом. Таким образом, роль управления проектом это создание условий при которых проект будет успешно реализован в кратчайшие сроки, при минимизации стоимости и контролируемых рисках на всех этапах разработки.

Тройственная ограниченность или треугольник управления проектом говорит о том, что достичь всех трех показателей: качественно, быстро и дешево, невозможно. Соответственно задачей является не достижение всех трех, а баланс, при котором проект можно назвать успешным.

Рисунок 1.1.1 - Иллюстрация тройственной ограниченности.

Примечание - собственная разработка.

Целью реализации любого проекта является достижение какого-то результата. В некоторых случаях результат может отличаться от того что планировалось изначально, иногда такие проекты тоже могут быть успешными. Например в процессе реализации могут измениться требования - мерой успешности в данном случае будет то, насколько быстро и безболезненно проект может быть адаптирован к этому и в конце концов реализован. Проект может быть признан неактуальным и закрыт на каком-то этапе - в этом случае мерой успешности будет минимизация издержек. Провальный проект может дать нам полезный опыт или новую технологию, иногда это может стать основой для будущих успешных проектов. Из всего этого следует значение управления проектом - любой проект должен нести в себе какую-то ценность, давать какой-то результат.

1.2. Виды методологий управления проектами по продуктовой разработке мобильных приложений

Waterfall

Waterfall - каскадная модель или “водопад”. Является одной из старейших существующих концепций организации технологического процесса разработки. История этого подхода берет начало в 1956 году [2]. Сам термин Waterfall был впервые использован в 1976 году [3]. Этот подход лег в основу военного стандарта “Defense system software development” министерства обороны США от 1985 года. Каскадная модель повсеместно использовалась и используется в совершенно различных сферах, таких как строительство, военная промышленность. Благодаря тому что она отлично формализуется она до сих пор занимает значимое место, особенно при работе с государственными структурами и крупными компаниями, для которых формализация и документация процессов является первоочередной целью, даже в ущерб срокам/стоимости/качеству продукта.

Выделяются шесть последовательных фаз (в некоторых источниках может быть пять):

- определение требований: на данным этапе составляется ТЗ;
- анализ, проектирование: результатом являются модели, схемы, бизнес правила;
- конструирование: разработка архитектуры программного решения;
- воплощение: разработка, релиз продукта;
- тестирование: проверка, отладка дефектов;
- поддержка и сопровождение.

Отличительной особенностью подхода является то, что переход к следующей фазе происходит только после завершения предыдущей, но возможны некоторые отклонения, например возврат к предыдущим фазам в случае возникновения проблем или обнаружения недостатков.

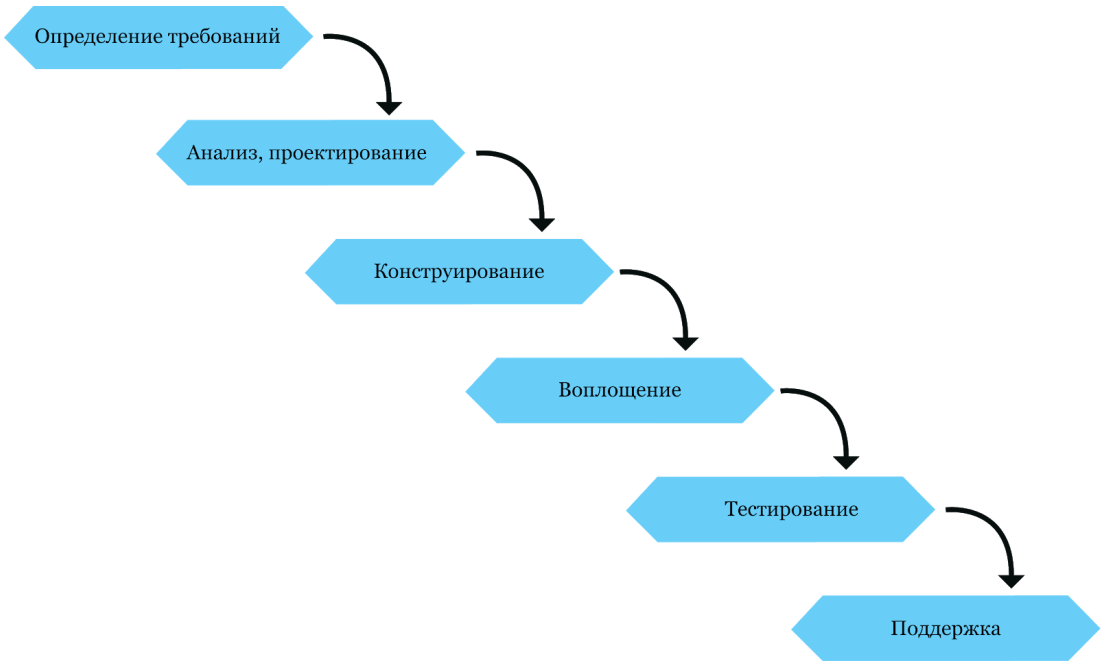

Рисунок 1.2.1 - Каскадная модель (“водопад”).

Примечание - собственная разработка.

Преимущества:

- время и ресурсы потраченные на начальных стадиях позволяют сэкономить на последующих стадиях, предусмотреть возможные проблемы заранее;
- процессы понятны и четко определены;
- каждая стадия как правило может быть хорошо документирована;
- заранее определены этапы процесса разработки;
- проще прогнозировать сроки и стоимость реализации (в сравнении с другими моделями).

Недостатки:

- если заказчик не имеет четкого представления о конечном виде продукта, сложность второго-третьего этапов возрастает, так как необходимо предусмотреть гибкость/расширяемость архитектуры;
- работающий продукт появляется на поздних этапах;
- в случае изменения требований к продукту, необходим возврат к начальным стадиям, что очень сильно увеличивает стоимость, особенно если это происходит на поздних стадиях;
- высокие риски и неопределенность на начальных фазах;

Modified Waterfalls

Modified Waterfalls - модифицированные водопадные модели. Как результат наличия недостатков водопадной модели существует ряд модифицированных водопадных моделей, которые учитывают данные недостатки (Sashimi - Waterfall with Overlapping Phases, Waterfall with Subprojects, Waterfall with Risk Reduction) [5]. Оставим за рамками данного исследования сами модели, хочется подчеркнуть только то, что все модифицированные модели построенные на основе водопадной решают проблемы “негибкости” водопадной модели:

- привносят возможность более раннего реагирования на обнаруженные проблемы, либо изменения требований;
- разделение проекта на меньшие (хотя это влечет за собой усложнение на этапе проектирования и необходимость интеграции);
- спиральный подход к фазам водопадной модели, в первую очередь применяется для первого этапа сбора требований, однако может быть применен для любой фазы.

Iterative and Incremental development

Iterative and Incremental development - итеративно-инкрементная модель. Основная концепция - поэтапная реализация проекта с непрерывным анализом полученных результатов полученных на предыдущих этапах. Каждый этап представляет из себя цикл (PDSA - Plan-Do-Study-Act) Планирование - Реализация - Проверка - Оценка.

Сам цикл PDSA был введен еще в 1930-х годах Вольтером Стюартом [7]. Активное развитие данная модель получила в 1970-х и 1980-х годах. Одним из первых применений данной модели является проект X-15 - гиперзвуковой самолет, хоть это и не был программный продукт он стоит упоминания. Часть команды работавшая над ним затем работала в NASA над программным обеспечением для пилотируемой космической программой США “Меркурий”, конкурирующей в то время с советской космической программой. При разработке проекта “Меркурий” использовалась

итеративно-инкрементная модель с итерациями по пол дня, а также использовались практики экстремального программирования. В дальнейшем практики этой модели были привнесены в компанию IBM, где применялись для разработки больших, критичных системам для министерства обороны. Например командно-управляющая система для подводных лодок, которая состояла из миллиона строк кода, была первым известным и документированным проектом применившим итеративно-инкрементную модель [8].

В дальнейшем она использовалась многими другим компаниями для разработки совершенно разных типов проектов и дала толчок для формирования других моделей и методологий. В Сингапуре создавалась крупная логистическая система, использовалась водопадная модель и проект испытывал проблемы с реализацией. Джефф де Лука пересмотрел процессы на этом проекте используя принципы итеративно-инкрементной модели и как результат предложил подход названный Feature Driven Development - разработка делающая акцент на функционале (управляемая функционалом). Применение модели в компании Easel Corporation привело к созданию Scrum методологии [8].

В 2000-м году вышел новый стандарт разработки программного обеспечения министерства обороны США, в котором рекомендовалось использовать эволюционное развитие и принципы итеративно-инкрементной модели для управления проектами [6].

Данный метод имеет две характеристики, которые отличаются представлением о том, какой результат получается на каждом этапе. С точки зрения итеративности - итоговый результат каждого цикла должен представлять целостную систему, которая будет улучшаться и дополняться с течением времени. Такой подход очень удобен с точки зрения владельца продукта - уже после первых итераций можно видеть целостную картину. С точки зрения инкрементности - продуктом каждого цикла является версия с добавленной частью нового функционала. О одной стороны это является плюсом - все усилия могут быть сконцентрированы на этой части, что позволяет выполнить работу быстрее и качественней, однако поскольку при таком подходе какой-то функционал остается нереализованным до последнего момента это может нести дополнительные риски.

Рисунок 1.2.2 - Итеративно-инкрементная модель.

Примечание - собственная разработка.

Преимущества:

- подходит для очень сложных проектов, для которых можно сформулировать предварительные требования, но неясны окончательные требования, либо не ясны способы реализации;
- первая рабочая версия продукта (или прототип) может быть выпущен уже после первого цикла;
- новые версии могут выпускаться после каждого цикла, что является плюсом для бизнеса и пользователей;
- данный подход позволяет более гибко реагировать на рост (либо наоборот сокращение) работ над проектом. При старте нового цикла команда может изменяться под текущие потребности проекта;
- позволяет приоритезировать реализацию различных направлений проекта;
- возможность использовать опыт полученный на предыдущих этапах.

RAD

RAD (от англ. 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 нечетко определенных требований к продукту. Эффект ускорения разработки достигается путем использования соответствующих технических средств и непрерывного, параллельного с ходом разработки, уточнения требований и оценки текущих результатов с привлечением заказчика. RAD создана в конце 1980-х как альтернатива более ранним каскадной и итеративной моделям. С конца XX века RAD получила широкое распространение.

Тот же термин используется в отношении программных инструментов быстрого прототипирования и разработки ПО. Типичными качествами таких инструментов являются максимальная автоматизация рутинных операций и широкое использование визуального программирования.

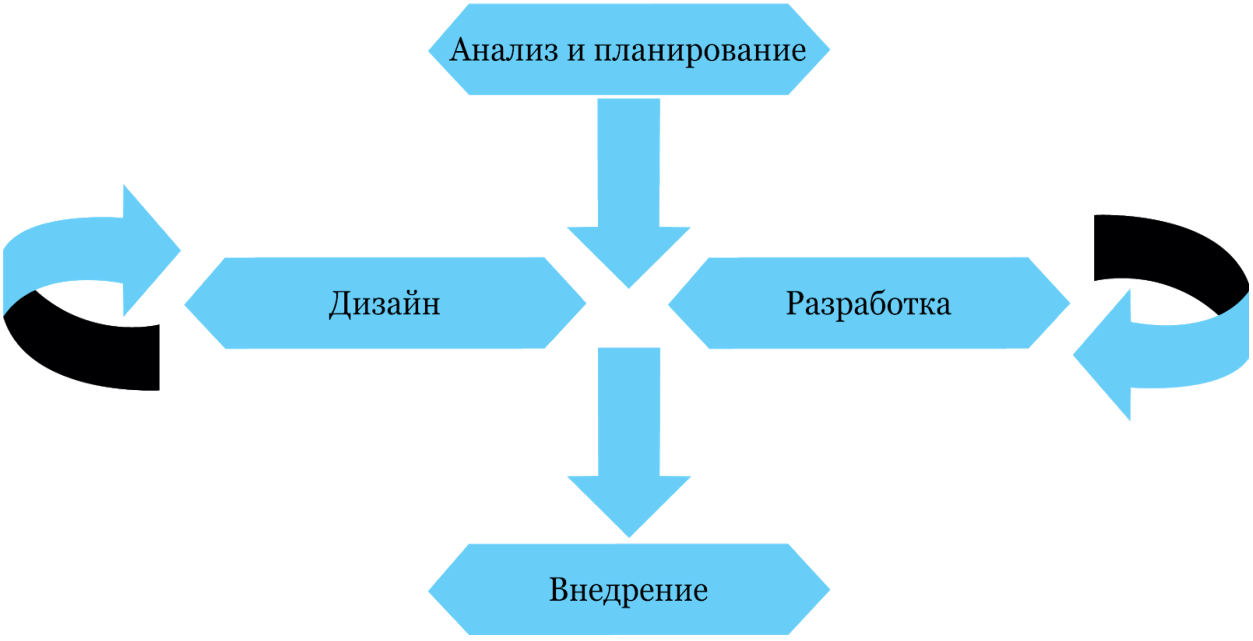

Рисунок 1.2.3 - RAD, быстрая разработка приложений.

Примечание - собственная разработка.

Методология может использоваться только при наличии высококвалифицированных и узкоспециализированных специалистов. Бюджет проекта должен быть достаточно большим, чтобы оплатить этих специалистов. RAD-модель может быть выбрана при уверенном знании

целевого бизнеса и необходимости срочного производства системы в течение 2-3 месяцев.

Преимущества:

- быстрота продвижения программного продукта на рынок;
- интерфейс, устраивающий пользователя;
- легкая адаптируемость проекта к изменяющимся требованиям;
- простоту развития функциональности системы.

Недостатки:

- если пользователи не могут постоянно участвовать в процессе разработки - это может негативно сказаться на результате;
- слабый контроль за процессом;
- архитектурные проблемы - фокусирование на результате и изменениях может сказаться на общей архитектуре решения в негативном плане;
- невозможность масштабируемости - не подходит для крупных проектов.

Scrum

Одна из самых распространенных современных методологий - Scrum. Scrum интересен в первую очередь тем, что он достаточно четко описывает как должны организовываться процессы внутри команды, содержит ряд правил, которые стандартизируют работу команд, что позволяет проще вливаться новым членам. Методологию Scrum можно назвать гибридом, который получился за счет применения подходов IID и ценностей, которые впоследствии стали частью Agile. С одной стороны четко регламентируется процесс организации спринтов, с другой он следует ценностям и принципам Agile, в то же время задает ряд правил и ритуалов для команды, чтобы организовать взаимодействие.

Скрам (англ. scrum «схватка») - подход к разработке программного обеспечения, который набор принципов, ценностей, политик, ритуалов, артефактов, на которых строится процесс SCRUM-разработки, позволяющий в жёстко фиксированные и небольшие по времени итерации, называемые

спринтами (sprints), предоставлять конечному пользователю работающий продукт с новыми бизнес-возможностями, для которых определён наибольший приоритет.

Кен Швабер и Джефф Сазерленд работали над Скрамом вплоть до 1995 года, когда они выступили с докладом на конференции OOPSLA (Object-Oriented Programming Systems, Languages and Applications) [14]. В этом докладе были отражены знания, полученные ими за прошедшие годы, и было впервые дано формальное определение Скрама [15]. Руководство по Скраму описывает фреймворк в том виде, в котором он был разработан и дополнялся Джеффом Сазерлендом и Кеном Швабером на протяжении более чем двадцати лет.

Рисунок 1.2.4 - Скрам процесс.

Примечание - собственная разработка.

Кен Швабер и Джефф Сазерленд работали над Скрамом вплоть до 1995 года, когда они выступили с докладом на конференции OOPSLA (Object-Oriented Programming Systems, Languages and Applications) [14]. В этом докладе были отражены знания, полученные ими за прошедшие годы, и было впервые дано формальное определение Скрама [15]. Руководство по Скраму описывает фреймворк в том виде, в котором он был разработан и

дополнялся Джеффом Сазерлендом и Кеном Швабером на протяжении более чем двадцати лет.

Согласно руководству по Скраму [15], определение самого Скрама звучит так: Скрам - это фреймворк, который помогает решать изменяющиеся в процессе работы задачи, чтобы продуктивно и творчески поставлять клиентам продукты с максимально возможной ценностью.

Примечание. Фреймворк (англицизм, неологизм от framework «остов, каркас, структура») — заготовки, шаблоны для программной платформы, определяющие архитектуру программной системы; программное обеспечение, облегчающее разработку и объединение разных модулей программного проекта. В контексте применения к Скраму следует трактовать как: набор базовых элементов и правил, своего рода каркас, на котором строится процесс разработки.

Скрам является:

- компактным;
- простым для понимания;
- трудным для совершенного овладения.

Скрам – это процессный фреймворк, который начали использовать для управления работой над сложными продуктами в начале девяностых годов. Скрам не является процессом, техникой или исчерпывающим методом. Напротив, Скрам — это фреймворк, в котором можно использовать разнообразные процессы и методы. Скрам проявляет несовершенства в управлении продуктом и методах работы, чтобы вы могли постоянно улучшать продукт, команду и рабочее окружение.

Основными элементами фреймворка являются Скрам-команды и связанные с ними роли, события, артефакты и правила. Каждый элемент фреймворка служит определенной цели и является обязательным для успешного использования Скрама.

Правила Скрама связывают вместе события, роли и артефакты, регулируют отношения и взаимодействия между ними. Они описаны в Руководству по Скраму [15].

Скрам был изначально разработан для управления продуктами и их разработки. С начала девяностых Скрам активно используется по всему миру, чтобы:

- исследовать и выявлять жизнеспособные рынки, технологии и возможности продуктов;
- разрабатывать продукты и улучшать их;
- выпускать продукты и их обновления по несколько раз в день;
- разрабатывать и поддерживать облачные технологии (онлайн, безопасно, по требованию) и другие среды для использования продуктов;
- поддерживать и обновлять продукты.

Скрам используется для разработки программного обеспечения, оборудования, встроенного программного обеспечения, автономных транспортных средств и микробиологических исследований. Скрам использовался в школах, правительстве, маркетинге, в управлении организациями — повсеместно и повседневно в жизни отдельных людей и сообществ.

В современном мире резко возросли сложность технологий, поведения рынков и окружающей среды, а также их взаимодействие. В этих условиях сложности и неопределенности Скрам ежедневно подтверждает свою пользу и необходимость применения.

Скрам доказал свою особую эффективность в итеративной и инкрементальной передаче знаний. Он широко используется в работе над продуктами, услугами и в управлении организациями.

Суть Скрама — это маленькая команда людей. Каждая отдельная команда гибка и адаптивна. Эти преимущества проявляются, распространяясь на любое количество команд в организации: одну, несколько или целые сети команд, которые разрабатывают, выпускают, осуществляют эксплуатацию и поддержку продуктов, таким образом объединяя труд тысяч людей. Они совместно работают и взаимодействуют благодаря продвинутым архитектурам и современным средам для выпуска.

Скрам основан на теории эмпирического управления (эмпиризме). Согласно этой теории, источником знаний является опыт, а источником решений – реальные данные. Скрам использует итеративный и

инкрементальный подход, чтобы улучшать прогнозируемость и управлять рисками. Процесс эмпирического управления основан на «трех китах»: прозрачности, инспекции и адаптации.

Прозрачность означает, что значимые характеристики процесса должны быть известны тем, кто отвечает за его результат. Прозрачность требует: эти характеристики должны быть обозначены общими соглашениями, чтобы все участники одинаково понимали происходящее. Например терминология должна быть общей для всех участников, критерии приемки должны быть общими для тех кто выполняет работу и тех, кто принимает результат.

Инспекция предполагает что все члены процесса должны регулярно исследовать артефакты Скрама и свой прогресс в движении к целям Спринта, чтобы вовремя обнаружить нежелательные отклонения. Частота проведения при этом должна выбираться таким образом, чтобы не мешать работе.

Адаптация говорит о необходимости изменения процессов если в результате инспекции выясняется, что одна или несколько характеристик процесса выходят за допустимые пределы, и это приводит продукт в неприемлемое состояние. Чем раньше будут внесены изменения, тем меньше риск дальнейших отклонений.

Скрам выделяет четыре формальных события для инспекции и адаптации:

- планирование Спринта;
- ежедневный Скрам;
- обзор Спринта;
- ретроспектива Спринта.

Согласно идеологии Скрама, когда Скрам-команда опирается на ценности Скрама (преданность, смелость, сфокусированность, открытость и уважение) и разделяет их, “три кита” фреймворка — прозрачность, инспекция и адаптация — реализуются и создают атмосферу всеобщего доверия. Участники Скрам-команды исследуют и постигают эти ценности по мере работы с событиями, ролями и артефактами Скрама. Руководство по Скраму постулирует, что успешность использования Скрама напрямую зависит от того, насколько хорошо люди придерживаются этих ценностей. Каждый участник должен быть предан целям Скрам-команды, сфокусирован на них и на их достижении в рамках Спринта. Все обладают смелостью

действовать правильно и работать над решением сложных задач. Заинтересованные лица и Скрам-команда соглашаются быть открытыми друг с другом в работе, несмотря на возникающие трудности, уважают профессионализм и самостоятельность друг друга. Наличие идеологии с такими утверждениями, большинству из которых не приводится объяснений и доказательств в Руководстве по Скраму, часто приводит к сравнению Скрама с религией - они должны приниматься членами команды как истина и утверждается, что благодаря следованию им, команда сможет достигать результатов. В этом есть слабость Скрама. В бюрократизированных и компаниях с жесткой организацией данная методология управления проектами едва ли будет применима в существующем виде.

Скрам-команда состоит из Владельца Продукта, Команды Разработки и Скрам-мастера. Скрам-команды являются самоорганизующимися и кросс-функциональными. Самоорганизующиеся команды самостоятельно решают, как выполнять свою работу, а не следуют внешним указаниям. Кросс-функциональные команды обладают всеми необходимыми компетенциями для выполнения работы и не зависят от людей, которые не входят в команду. Модель команды в Скраме направлена на улучшение гибкости, творчества и продуктивности. Скрам-команды поставляют продукт итеративно и инкрементально, максимально используя возможности для получения обратной связи. Благодаря тому, что Готовый продукт поставляется инкрементами, работоспособная и потенциально полезная версия продукта доступна в любой момент.

Владелец Продукта несет ответственность за достижение максимальной ценности продукта как результата работы, которую выполняет Команда Разработки. Способы достижения максимальной ценности могут различаться и зависят от самих организации, Скрам-команд и конкретных людей. Владелец Продукта – единственный человек, который отвечает за управление Бэклогом Продукта. Управление Бэклогом Продукта включает в себя:

- описание Элементов Бэклога Продукта ясным и понятным образом;
- управление порядком Элементов Бэклога Продукта для наилучшего достижения целей и миссий;
- оптимизацию ценности работы, выполняемой Командой Разработки;

- обеспечение доступности, прозрачности и ясности Бэклога Продукта для всех участников процесса. Бэклог Продукта при этом отражает, над чем Скрам-команда будет работать дальше;
- гарантию что Команда Разработки в достаточной степени понимает Элементы Бэклога Продукта.

Владелец Продукта может выполнять эту работу как самостоятельно, так и делегировать её выполнение членам Команды Разработки. Тем не менее ответственность за Бэклог Продукта лежит на плечах Владельца Продукта. Роль Владельца Продукта исполняет один человек, а не группа людей. Владелец Продукта может отражать пожелания заинтересованных лиц в Бэклоге Продукта, но любой кто хочет изменить приоритет элемента, должен обращаться к Владельцу Продукта. Для успешного выполнения обязанностей Владельцем Продукта все сотрудники организации должны уважать его или её решения. Решения, которые принимает Владелец Продукта, отражены в содержании и порядке Элементов Бэклога Продукта. Скрам указывает, что никто не может заставить Команду Разработки работать над другим набором требований.

Команда Разработки состоит из профессионалов, которые работают над поставкой к концу Спринта готового к выпуску Инкремента Продукта. Инкремент должен быть готов к Обзору Спринта. Созданием Инкремента занимаются только участники Команды Разработки. Организации формируют Команды Разработки и наделяют их всеми полномочиями для самостоятельной организации и управления своей работой. Команды Разработки обладают рядом характеристик:

- самоорганизованность: никто (даже Скрам-мастер) не может указывать Команде Разработки как превратить Бэклог Продукта в готовые к релизу Инкременты;
- кросс-функциональность: команда обладает всеми навыками, которые необходимы для создания Инкремента Продукта;
- разработчик — единственная роль для членов Команды Разработки, независимо от типа задач, которые он выполняет. Скрам не признает других ролей в Команде Разработки, это правило не имеет исключений;
- скрам не признает подкоманд в Команде Разработки, независимо от областей, над которыми необходимо работать (например, тестирование,

архитектура, эксплуатация или бизнес-аналитика). Это правило не имеет исключений;

- команда Разработки несет коллективную ответственность за создание Инкремента Продукта. При этом отдельные члены Команды Разработки могут обладать различными специализированными навыками и экспертизой.

Скрам не дает однозначного ответа относительно оптимального размера Команды Разработки, но декларирует, что он должен быть достаточно мал, чтобы команда оставалась гибкой, и достаточно велик, чтобы выполнять значимую работу за время Спринта. Согласно разным источникам идеальный размер - 5-7 человек [26]. При этом в Руководстве по Скраму на основании эмпирических данных говорится следующее. Когда в Команде менее трех человек, взаимодействие и производительность снижается. Небольшие Команды Разработки могут столкнуться с проблемой нехватки навыков для создания готового к выпуску Инкремента Продукта. Команды размером более девяти человек испытывают трудности с координацией работы. Сложность работы в больших Командах Разработки возрастает настолько, что эмпирический процесс становится неприменим. Не учитываются при подсчете численности Команды Разработки, Владелец Продукта и Скрам-мастер, если они сами не выполняют работу из Бэклога Спринта.

Обязательные события Скрама предусмотрены, чтобы процесс был регулярным, и другие собрания были бы не нужны. Каждое событие ограничено по времени и не может превышать максимальную установленную длительность. К событиям относятся: спринт, планирование спринта, ежедневный скрам, обзор спринта, ретроспектива спринта.

Артефакты Скрама отражают работу или ценность. Они обеспечивают прозрачность и создают новые возможности для инспекции и адаптации. Артефакты Скрама специально разработаны для обеспечения максимальной прозрачности ключевой информации, чтобы все участники процесса обладали её одинаковым пониманием. К ним относятся: бэклог продукта, бэклог спринта, инкремент.

Детальное описание фреймворка представлено в Руководстве по Скраму[15]. Роли, артефакты, правила и события Скрама не подлежат изменению. Хотя использование отдельных элементов данного фреймворка допустимо, но полученный результат не может называться Скрамом. Скрам

существует только в качестве цельного фреймворка, но он может вмещать в себя другие техники, методологии и практики.

Преимущества:

- возможность быстрого запуска проекта с наиболее приоритетными функциями и минимально возможным бюджетом;
- ежедневный контроль над ходом работ, и более гибкий контроль над бюджетом проекта;
- частые демонстрации проекта. Применение данной методологии предполагает регулярную демонстрацию разработок заказчику, что позволяет в будущем избежать полного провала работы команды и разочарований клиента;
- возможность вносить коррективы в техническое задание по ходу реализации проекта, что является несомненным преимуществом для владельца продукта (заказчика).

Недостатки:

- юридические и коммерческие аспекты - в случае выполнения работ для стороннего заказчика невозможно зафиксировать стоимость работ по проекту;
- неприменима для работы с сильно бюрократизированными структурами и организациями;
- сложности применения при низкой квалификации и не мотивированности команды на получении результатов;
- слабо масштабируется сам подход, в случае работы над очень большим проектом приходится разбивать проекты на модули.

Agile

Agile - гибкая модель. Подход к разработке программного обеспечения при котором требования и решения эволюционируют благодаря совместным усилиям самоорганизованной и самодостаточной команды и конечным пользователем (или заказчиком). Подход пропагандирует адаптивное планирование, эволюционную развитие, раннюю доставку продукта,

последовательное улучшение, а также стремление к быстрому и гибкому реагированию на изменения.

В 2001 году, в городе Сноуберд, штат Юта, произошла встреча семнадцати разработчиков программного обеспечения. В их числе были разработчики фреймворка Scrum Джефф Сазэрлэнд и Кен Швайбер. Целью встречи было обсуждение “легковесных” подходов к разработке программного обеспечения, таких как RAD, UP, DSDM, Scrum, Crystal Clear, XP. Результатом встречи стал Agile-манифест разработки программного обеспечения (Manifesto for Agile Software Development) [13]. После этого активно начал использоваться сам термин Agile. Agile-манифест - набор ценностей и принципов гибкой разработки программного обеспечения. Ценности и принципы были почерпнуты из указанных выше “легковесных” подходов, таким образом Agile-манифест сконцентрировал их в себе и стал символом тенденций последних десятилетий в управлении разработкой программного обеспечения.

Манифест объявляет следующие ценности:

- 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.

“То есть, не отрицая важности того, что справа, мы все-таки больше ценим то, что слева” - гласит комментарий манифеста. Взглянем на эти пункты детально - какие цели стоят за этими ценностями.

Поскольку мы говорим про сферу разработки программного обеспечения, сотрудники - это люди умственного труда, для них важна комфортная обстановка в коллективе и мотивация, иначе высока вероятность того, что производительность труда будет низкая, либо сотрудник покинет команду.

Другая особенность в том, что хорошо написанный код, в принципе сам может являться документацией. Также существуют решения для генерации документации на основании написанного кода. Соответственно документация появляется в процессе, нет необходимости делать

документацию на те задачи, которые пока не планируются/делаются/не будут сделаны никогда.

Сотрудничество с заказчиком позволяет решать проблемы проекта гораздо быстрее; иногда потенциальные проблемы реализации могут быть решены до самого момента реализации - достаточно комментариев или уточнений со стороны команды; оценка задач происходит до момента когда принимается решение о начале выполнения, соответственно заказчик может принять решение о приоритетности не только на основании важности задачи, но и на основании сложности ее выполнения (очень часто происходит так, что после оценки задача либо получает очень низкий приоритет и есть шанс что она никогда не будет реализована, либо отвергается в принципе).

Краеугольный камень данной модели - любая задача (процесс) должна решать какую-то проблему пользователя. Сервис Pivotal Tracker, который часто используется Agile командами, рекомендует при составлении описания задач руководствоваться именно этим принципом - основной упор делается на бизнес-кейс и условия приемки, которые описываются так: контекст - действие - реакция [17]. Если приложение или сервис не решает проблему пользователя - следует пересмотреть задачу, либо отказаться от реализации. Если мы ошиблись с оценкой - надо ее пересмотреть и изменить план, иначе мы либо не укладываемся в сроки, либо пытаемся уложиться в план - перегружаем команду работой, что влечет за собой негативные последствия.

Основополагающие принципы Agile-манифеста:

- наивысшим приоритетом для нас является удовлетворение потребностей заказчика, благодаря регулярной и ранней поставке ценного программного обеспечения;
- изменение требований приветствуется, даже на поздних стадиях разработки;
- Agile-процессы позволяют использовать изменения для обеспечения заказчику конкурентного преимущества;
- работающий продукт следует выпускать как можно чаще, с периодичностью от пары недель до пары месяцев;
- на протяжении всего проекта разработчики и представители бизнеса должны ежедневно работать вместе;

- над проектом должны работать мотивированные профессионалы. Чтобы работа была сделана, создайте условия, обеспечьте поддержку и полностью доверьтесь им;
- непосредственное общение является наиболее практичным и эффективным способом обмена информацией как с самой командой, так и внутри команды;
- работающий продукт — основной показатель прогресса;
- инвесторы, разработчики и пользователи должны иметь возможность поддерживать постоянный ритм бесконечно. Agile помогает наладить такой устойчивый процесс разработки;
- постоянное внимание к техническому совершенству и качеству проектирования повышает гибкость проекта;
- простота — искусство минимизации лишней работы — крайне необходима;
- самые лучшие требования, архитектурные и технические решения рождаются у самоорганизующихся команд;
- команда должна систематически анализировать возможные способы улучшения эффективности и соответственно корректировать стиль своей работы.

Исходя из ценностей и принципов, Agile – это философия организации рабочего процесса. Она не решает конкретные задачи организации рабочих процессов, зато даёт площадку для формирования важных для заказчика продуктов в максимально короткие сроки. В первую очередь, Agile необходима в сферах, где происходит разработка новых продуктов. Чаще всего, это разработка программного обеспечения или исследовательские проекты, но список можно продолжать и дальше – результатом работы должен может стать любой процесс, если виден его конечный продукт. Ключевая особенность Agile – ценный результат – стирает ограничения в сферах использования методологии. Есть примеры, когда её не менее эффективно применяют в маркетинге или рекрутинге.

Рисунок 1.2.5 - Agile.

Примечание - собственная разработка.

Agile-манифест не содержит практических советов по работе, это только установки ценностей и принципов. Во многих статьях происходит сравнение Agile и “каскадной модели”, что на мой взгляд не всегда корректно, потому что сама по себе Agile модель не описывает процессов, при должном рвении можно адаптировать “каскадную модель” соответствовать принципам Agile и создать еще одно модифицированную каскадную модель: Waterfall with Agile Phases.

Существует большое количество методологий, которые объединяются в общую группу Agile: RAD, Scrum, XP, FDD, Lean, DSDM, Crystal. Эти методологии объединяют принципы и ценности Agile. При этом большинство этих методологий были разработаны гораздо раньше, чем были сформулированы принципы Agile. Таким образом, все эти подходы и их ценности стали основой для этой модели. В некотором смысле Agile модель является вершиной эволюции идей и подходов к управлению проектами в сфере разработки программного обеспечения. То что основополагающим принципом является гибкость - неудивительно, так как наиболее общей характеристикой всех проектов связанных с разработкой программного

обеспечения является неопределенность и риски. Гибкость позволяет реагировать на сложности и делать это только в тот момент, когда это необходимо, избегая необходимости учитывать это на ранних стадиях расходуя на это время и ресурсы. Сама Agile модель при этом не является руководством к действию, не описывает процессов и не помогает организовать систему управления проектом в прямом смысле.

1.3. Основные подходы к оценке эффективности управления проектами по продуктовой разработке мобильных приложений

При оценке эффективности управления проектами необходимо рассматривать обширный набор аспектов-критериев. Существуют различные подходы к оценке эффективности управления проектами (Project Management Value), основывающиеся на методиках различных организаций (как коммерческих, так и независимых научных), оптимизированных для использования в разных областях хозяйственной деятельности.

Оценка эффективности основывается на определении, выборе критериев для рассмотрения и оценки системы по этим качествам. Набор критериев может зависеть от сферы деятельности организации, характеристики проектов и состава системы.

Критерии, показатели и оценки можно условно разделить на две группы: качественные и количественные. Количественные оценки дают легко осязаемый, наглядный показатель эффективности, однако не всегда дают полное представление о всех преимуществах использования системы управления проектами.

Одна из методологий качественной оценки эффективности основана на экспертной оценке Критических факторов успеха (КФУ), выполнение которых необходимо для успешной реализации проекта. Механизм для стратегической оценки проекта в целом, основанный на экспертной оценке. Данный метод рекомендуется использовать неоднократно на этапе выполнения проекта. Его проводят циклически – через определенные промежутки времени, например, каждый месяц или при закрытии этапа проекта.

Основные проблемы оценки эффективности управления проектами по разработке мобильных приложений связаны с тем, что проекты сложно

стандартизировать. Использование методологий относящихся к Agile делает эту оценку еще сложнее. Выходом в этой ситуации является правильный выбор KPI (англ. Key Performance Indicators, KPI) - ключевых показателей эффективности которые лучше всего характеризуют эффективность работы проекта. Для мобильных приложений это могут быть:

- стабильность приложения (% сессий без крашей);
- количество пользователей;
- коэффициент удержания (% возврата пользователей в приложение спустя определенный период времени);
- процент успешно реализованных фич;
- время реализации нового функционала с момента возникновения идеи и до момента доставки его пользователям.

Разнохарактерность сферы мобильных приложений приводит к тому, что не существует универсального метода оценки эффективности. Каждый проект индивидуален и требует разработки собственной системы оценки эффективности, что требует эмпирических знаний. На примере реально действующих проектов, которые будут рассмотрены далее в разделе 2.3, можно выделить такие количественные KPI:

- срок реализации проекта;
- количество новых частей функционала реализованных в рамках данного проекта;
- стабильность приложения;
- количество багов (ошибок);
- рейтинг приложения в магазине.

Собирая статистику этих KPI по проектам и анализируя можно делать выводы о том насколько успешен был сам проект, улучшаются ли эти показатели от проекта к проекту, определить общие проблемы в существующих подходах к управлению проектами. В тестировании существует такой подход как A/B тесты. Он основан на использовании двух версий продукта или части его функционала и разделении аудитории которой они поставляются. Данный подход можно использовать и для оценки эффективности управления проектами. Используя разные методологии, либо

модифицируя часть процессов управления и оценивая одинаковый набор KPI можно достаточно объективно сравнивать их между собой и применять лучшие практики в будущих проектах.

ГЛАВА 2
АНАЛИЗ ЭФФЕКТИВНОСТИ УПРАВЛЕНИЯ ПРОЕКТАМИ ПО ПРОДУКТОВОЙ РАЗРАБОТКЕ МОБИЛЬНЫХ ПРИЛОЖЕНИЙ

2.1. Сравнительная характеристика программно-технического инструментария управления проектами по продуктовой разработке мобильных приложений

Специфика мобильной разработки заключается в том, что проекты не такие большие например по сравнению с энтерпрайз решениями, команды более 10 человек как правило являются исключениями. Эта особенность сказывается и на используемы инструментах для управления проектами. Наиболее популярными инструментариями для управления проектами по мобильной разработке в настоящее время являются JIRA, Trello, Pivotal Tracker, GitHub, Asana, Zenkit и множество других. Изначально эти инструменты позиционируются больше как инструменты для взаимодействия команды (англ. Team Collaboration Tools) или системы отслеживания ошибок, однако используются и для управления проектами. Основные функции всех этих инструментов (могут отличаться):

- управление задачами (список задач, приоритизация, история);
- отслеживание выполнения задач;
- назначение ответственных лиц за выполнение задачей;
- управление итерациями и сроками сдачи;
- анализ показателей продуктивности работы команды и отдельных членов;
- трекинг времени.

На данный момент практически все инструменты имеют интеграции с другими, что позволяет автоматизировать большое число процессов, а также использовать их для разного уровня планирования. Например можно использовать Trello для стратегического планирования глобальных задач, поскольку он максимально прост и позволяет легко визуализировать процесс,

а Pivotal Tracker для организации бэклога технических задач и отслеживания статуса их готовности.

JIRA - наиболее сложный и многофункциональный инструмент. Позволяет как управлять задачами, так и организовывать учет времени, что позволяет избежать необходимости использования других инструментов для этого. Обладает большим количеством настроек для оптимизации процессов под определенные нужды. Очень гибка и имеет большое количество расширений. Подходит для проектов использующих различные методологии управления проектами. Например может быть адаптирована для Скрама или Канбана. Обладает интеграциями с другими сервисами.

Trello - сервис основанный на принципе использования Канбан досок для управления проектом. Максимально прост в использовании. Процесс работы визуализирован. Является практически идеальным инструментом для маленьких команд. Плохо масштабируется, при росте команды и увеличении сложности проекта может потребоваться переход на какую-то другую систему. Может быть использован как дополнительный инструмент при использовании других инструментов управления проектом, например для стратегического планирования. Простота, быстрота освоения и старта трелло-досок, позволяет использовать для каких-либо маленьких короткоживущих проектов внутри проектов.

Zenkit - альтернатива Trello, обладает более широким набором функционала и возможностей при сохранении простоты Trello. Позволяет использовать разные способы визуализации: канбан-доска, список, календарь, таблица. Очень хороший вариант управления проектом, если предполагается что проект может разрастись, но при этом хочется сохранить простоту Trello и не усложнять все (JIRA).

Pivotal Tracker — сервис, позволяющий команде быстрее идти к цели с автоматическим прогнозированием даты завершения, основанной на прошлых показателях. Поощряет постоянную обратную связь с клиентами и приоритеты по каждому проекту. Сервис умеет выдавать реалистичные оценки, предоставлять опции документооборота и менеджмента. Каждый проект находится в двух кликах от стартового экрана, имеется также полнотекстовый поиск, с поддержкой более мощных запросов. При этом инструмент никогда не позволяет разработчикам упускать из виду как большие проекты, так и самые мелкие рабочие детали. Pivotal Tracker также

обладает открытым API, который уже интегрирован со многими популярными сторонними инструментами. Удобен при наличии взаимосвязанных задач - можно создавать связи между задачами, определять последовательность выполнения, блокеры и агрегировать некоторое количество задач в одну. Основные характеристики веб-сервиса:

- добавление задач и комментариев;
- отслеживание задач и распределение первоочередных приоритетов;
- работа в режиме реального времени, управление релизами и др;
- удобен для гибкого итеративного управления рабочим процессом;
- предоставляет удобную возможность планирования итерации на основании оценки сложности задач;
- анализ производительности команды;
- интеграции со сторонними механизмами.

Сервис удобен в первую очередь для команд работающих по моделям имеющих в своей основе итеративный подход, также удобен для различного рода Agile команд.

GitHub в первую очередь используется как веб-сервис для хостинга проектов основанный на системе контроля версий Git. Содержит систему управления задачами и багтрекинга. Однако она достаточно гибкая, чтобы предоставлять эффективную систему управления целыми проектами, большими и маленькими. В GitHub есть задачи со взаимными ссылками, теги для прикрепления метаданных к вашим задачам, методы для прикрепления кода к задачам и даже возможность создания ключевых событий для группировки задач внутри фаз разработки проекта. В первую очередь ориентирован на разработчиков. Обычно используется для проектов с открытым кодом, поддерживаем сообществом разработчиков. Существуют системы, которые могут интегрироваться с GitHub и позволяют предоставляют возможности организовать более удобное представление для управления задачами, но в чистом виде является неудобным инструментом именно для управления, поэтому и остается больше инструментом для разработчиков.

2.2. Анализ методологий управления проектами по продуктовой разработке мобильных приложений

Одной из задач работы является сравнение эффективности разных методологий для реализации проекта. Мы составим методику оценки сроков и стоимости реализации проекта с учетом фактора неопределенности. Ключевая идея в том, что это не попытка сделать инструмент для оценки, скорее ретроспективное сравнение разных типов методологий, поэтому необходим ряд допущений. Одним из основных допущений примем то, что сама реализация проекта в коде неизменна при разных методологиях, то есть в итоге мы получаем одинаковый продукт (с идентичным функционалом), который реализован одинаковыми инструментами, сотрудниками одинаковой квалификации и используются одинаковые архитектурные подходы для решения, как результат и объем проекта в коде примерно идентичен по объему. Другое допущение - цена времени наших сотрудников равноценно и равна 1 в неделю.

Исследование проведем на примере некоего усредненного небольшого проекта. Это довольно типичная ситуация для сферы мобильной разработки. Определимся с характеристиками проекта, учтем ряд событий которые часто случаются:

1. на момент начала проекта есть концепт;
2. нет проблем со сбором требований, итоговый функционал без проблем формализуется на момент старта проекта;
3. реализация проекта в коде требует ориентировочно по 2 мифических человеко-месяца на серверную часть и мобильный клиент;
4. некоторый процент функционала в процессе реализации требует изменений (еще раз подчеркну, что при старте проекта это не учитывается, на это не закладывается время и ресурсы).

Пояснения:

1. любой проект имеет в своей основе какую-то идею;
2. не рассматриваем случаи, когда необходимо длительное исследование рынка, либо требования меняются в процессе их сбора. В защиту

самого метода исследования хочется отметить, что в проблемы на данном этапе могут быть а) относительно легко разрешены при любом из подходов и б) примерно одинаково влияют на сроки/стоимость при любом из подходов;

3. это наше допущение, также для простоты будем считать что 2 человека сделают работу ровно в два раза быстрей (хотя это и не так, но при таком малом размере команды, данный фактор не столь значим);
4. конечно обычно такие вещи учитываются, но в нашем случае это не важно. Мы можем учесть что нам потребуется изменить 25% функционала, но по факту может оказаться что это число 50 или 100. Идея в том, чтобы понять как методология отреагирует на изменения, которые мы не можем прогнозировать.

Задачи:

1. составить методику оценки срока реализации проекта при разных методологиях;
2. составить методику оценки стоимости реализации проекта при разных методологиях;
3. оценить влияние фактора неопределенности на сроки и стоимость реализации.

Сравнивать будем “каскадную модель” и Scrum, как два наиболее простых и достаточно формализованных для сравнения подхода. Определимся с составом сотрудников.

Команда работающая по “каскадной модели” будет состоять из руководителя проекта, дизайнера, двух бэкэнд и двух мобильных разработчиков, а также двух тестировщиков. Особенность этой модели предполагает что работа идет строго поэтапно, поэтому предположим что мы можем свободно осуществлять ротацию сотрудников и привлекать их только тогда, когда нам это необходимо, но учтем и “полную” стоимость, то есть стоимость времени всех сотрудников при условии что мы не можем этого делать и нам придется оплатить время всех сотрудников, которые будут заняты на проекте. Также для ускорения процесса мы будем привлекать по 2 сотрудника на этапах разработки и тестирования (за счет чего мы можем сократить время в 2 раза). Опять же для простоты предполагаем, что наши

сотрудники достаточно квалифицированы, чтобы отвечать за этапы 2 и 3 - проектирование и конструирование архитектуры. Расчет приведен в таблице 2.2.1:

Таблица 2.2.1 - Оценка сроков и стоимости выполнения проекта по каскадной модели.

Фазы:	Время, недель	Проектный менеджер	Дизайнер	Бэкэнд разработчик	Мобильный разработчик	Тестировщик	Стоимость, Итого:
1	1	1					1
2-3	1	1	1	1	1		4
4	4			8	8		16
5	1	1		1	1	2	5
6	1	1		1	1	1	4
<1-5>	7	3	1	10	10	2	26
Коррекция	1.4	0.6	0.2	2	2	0.4	5.2
Итого:	9.4	4.6	1.2	13	13	3.4	35.2
				“Полная” стоимость:			75.2

Примечание - собственная разработка.

Команда работающая по Скраму будет состоять из владельца продукта (в формулировке Скрама - Product Owner, PO) и собственно Скрам-команды, состав которой постоянен и неизменен на все протяжении проекта. Скрам-команда будет состоять из дизайнера на полставки, двух бэкэнд, двух мобильных разработчиков и тестировщика. По поводу дизайнера на полставки необходимо пояснение. Как правило роль дизайнера такова, что его постоянное присутствие на проекте не требуется, также его работа может быть растянута на некоторый период. Scrum методология не отрицает возможность работы человека в нескольких командах, основные условия в таких случаях - возможность человека соблюдать “ритуалы” Scrum, присутствовать на митингах, учавствовать в ретроспективах.

Данные в таблице 2.2.2 рассчитаны исходя из коэффициента коррекции равном 20% - процент изменений которые были внесены заказчиком во время реализации. Такого понятия как “Сбор требований, организационный” в скраме отсутствует, но хотелось бы как-то это учесть, поскольку до момента формулировки задач, их эстимации и поступления первых задач в работу проходит какое-то время, это не имеет особого значения для

долгосрочных проектов, но в нашем примере это будет влиятельным фактором. Время реализации в скраме увеличено на 25% (+1 неделя по сравнению с каскадной моделью, поскольку работа по четко определенным требованиям описанным в техническом задании в принципе проще).

Таблица 2.2.2 - Состав и стоимость работы Скрам-команды.

1	Сбор требований, организационный
5	Основная работа команды связанная с реализацией функционала 5 недель
0.6	дополнительные спринты - спринты с учетом коэффициента коррекций - изменений в проекте, которые производятся в процессе, рассчитаны по формуле <количество спринтов>* <коэффициент неотловленных коррекций>
Общее время:	6.6
Общая стоимость:	42.9

Примечание - собственная разработка.

Расчет коэффициента неотловленных коррекция для Скрама производится следующим образом:

- коэффициент отловленных коррекций до финального релиза рассчитывается как (1 - 1/количество спринтов);
- коэффициент эффективности работы Владельца продукта = 0.5 - коэффициент показывающий какое количество изменений Владельца продукта способен согласовать с заказчиком до момента реализации, основываясь на своем опыте/комментариях разработчиков/взаимодействии с заказчиком
- % отловленных коррекций до момента реализации = коэффициент отловленных коррекций до финального релиза * коэффициент эффективности работы Владельца продукта
- % измененных требований, скорректированных для Scrum = 20% изменений * (1 - % отловленных коррекций до момента реализации)

Значения по расчету приведены в таблице 2.2.3:

Таблица 2.2.3 - Расчет неотловленных коррекций.

Коэффициент отловленных коррекций до финального релиза:	0.8
Коэффициент эффективности работы владельца продукта:	0.5
% отловленных коррекций до момента реализации:	40.00%
% неотловленных коррекций:	12.00%

Примечание - собственная разработка.

На основании данной модели можно рассчитать время реализации, оценочную стоимость и полную стоимость проекта для каскадной модели и Скрама. В таблице 2.2.4 сведены итоговые результаты проведенного сравнения. Приведено общее время реализации проекта, оценочная и полная стоимости для двух методологий в зависимости от фактического количества изменений в проекте (коэффициент 0-1):

Таблица 2.2.4 - Сводная сравнительная таблица зависимости сроков и стоимости разработки проекта.

	Каскадная модель			Скрам		
Коррекция	Время	Стоимость, оценочная	Стоимость, полная	Время	Стоимость, оценочная	Стоимость, полная
0	8	30	64	6	39	39
0.1	8.7	32.6	69.6	6.3	40.95	40.95
0.2	9.4	35.2	75.2	6.6	42.9	42.9
0.3	10.1	37.8	80.8	6.9	44.85	44.85
0.4	10.8	40.4	86.4	7.2	46.8	46.8
0.5	11.5	43	92	7.5	48.75	48.75
0.6	12.2	45.6	97.6	7.8	50.7	50.7
0.7	12.9	48.2	103.2	8.1	52.65	52.65
0.8	13.6	50.8	108.8	8.4	54.6	54.6
0.9	14.3	53.4	114.4	8.7	56.55	56.55
1	15	56	120	9	58.5	58.5

Примечание - собственная разработка.

Полученные выводы о сравнении разных методологий по построенной методике:

1. стоимость реализации проекта по выбранным условиям согласно данной методики для Scrum оказывается выше, даже при условии внесения 100% изменений в продукт, но в целом цена реализации склонна увеличиваться меньшими темпами, нежели при использовании “каскадной модели”;
2. время реализации при использовании Scrum при условии отсутствия изменений меньше на 30% и с увеличением количества изменений Scrum эта разница только увеличивается. Это хорошо коррелирует с утверждениями, что гибкие методологии лучше приспособлены к изменяющимся требованиям;
3. дополнительно стоит обратить внимание на “полную” цену времени “каскадной модели”. В случае невозможности свободной ротации специалистов и необходимости оплачивать все их время цена реализации проекта по этой модели возрастает почти в два раза, что делает ее совершенно непригодной в некоторых случаях. С другой стороны, если проект совершенно не критичен ко времени, либо компания одновременно реализует множество проектов “каскадная модель” может даже позволить снизить стоимость реализации проектов.

Из изложенного выше следует, что при выборе методологии следует в первую очередь отталкиваться от особенностей проекта. Скрам (и другие модели Agile) хороши для долгосрочных проектов и проектов с высокой неопределенностью - на таких проектах может быть в полной мере раскрыта их эффективность. Для краткосрочных проектов, либо для проектов с четко определенными требованиями применение каскадной модели оправдано с точки зрения времени реализации и количества потраченных ресурсов. При этом необходимо учитывать, что при применении каскадной модели существуют “простои” - периоды, когда часть специалистов не может выполнять работы. Если избежать расходов, связанных с ними невозможно, например нанимая сотрудников только в момент необходимости, либо переключения их на другие проекты, то тогда ситуация меняется и полная стоимость ресурсов может значительно превысить оценочную.

2.3. Оценка эффективности управления проектами по продуктовой разработке мобильных приложений (на примере долгосрочных и краткосрочных проектов)

В качестве практического примера я бы хотел рассмотреть подход к управлению проектами используемый в компании, в которой я работаю последние два года. Мы занимаемся разработкой продуктов предоставляющих возможность поставлять динамический контент, а также осуществлять взаимодействие с конечными пользователями в режиме реального времени. Практически все решения основаны на одном базовом наборе функционала, который изменяется и/или дополняется под требования конкретного клиента. В основном наши приложения используются как дополнение к разным телевизионным шоу, но также есть продукты, которые выпадают из типичной картины. Отдельно существуют команды, которые работают над внутренними продуктами, которые являются долгосрочными и являются основой для клиентских решений. Для более правильного понимания я разделю все наши проекты на два типа: клиентские и внутренние, и рассмотрю их типичные характеристики.

Клиентские:

- основаны на базовом функционале, который работает “из коробки”;
- есть видение окнчательного состояние проекта;
- как правило короткоживущие. Этап активной разработки до 2 месяцев, период активного использования и поддержки 2-3 месяца;
- основная работа заключается в настройке и редизайне базового функционала;
- отсутствие высоких требований к решениям внедряемых для проекта (исключение - функционал, который предполагается использовать и на других проектах, то есть тот который должен попасть во внутренние проекты);
- наличие жестких дедлайнов.

Внутренние:

- собственные разработки;

- нечетко определенные требования к окончательному состоянию продукта;
- долгоживущие;
- основная работа это поддержка и внедрение новых функций;
- высокие требования к качеству решений, так как от них зависит большое количество клиентских проектов;
- отсутствие жестких дедлайнов, но могут быть ситуации, когда требования клиентских приложений могут эскалироваться на уровень внутренних проектов, что требует реприоретизации задач;
- окончательный вид продукта.

Для клиентских проектов отсутствует некая единая методология управления. Каждый член команды руководствуется своими наборами инструкций, устоявшимися правилами и личным видением ситуации. Для руководителя проекта это как правило набор руководств, составленных на основе имеющегося опыта. Для тех-лидов это также руководства составленные на основе опыта накопленного на предыдущих проектах, поскольку большая часть операций повторяется от проекта к проекту. Наиболее близким определением процесса разработки с точки зрения разработчиков будет RAD либо Канбан. Если попытаться охарактеризовать в целом имеющийся подход с точки зрения существующих методологий, то я бы назвал его хаотически-каскадная быстрая разработка приложений. Удивительно, но она работает - главное что клиент доволен. Причины почему работает:

- есть право на ошибку;
- отсутствие конкуренции;
- метрики задаются не конечным пользователем, а вернее отсутствие метрик и анализа;
- мотивированность членов команды;
- возможность ускорить процесс разработки за счет привлечения большего числа разработчиков;
- в исключительных случаях может происходить пересмотр задач с переносом на более поздние релизы, либо отказ от их реализации.

Причины почему процессы не налажены процессы внутри клиентских команд можно разделить на внешние и внутренние факторы. Рассмотрим их.

Экзогенные факторы:

- на момент старта проекта требования известны, но остаются не проработанные части функционала, которые обсуждаются и согласовываются позднее;
- дизайны могут быть окончательно утверждены уже на поздней стадии;
- в любой момент могут потребоваться изменения.

Эндогенные факторы:

- руководитель проекта как правило человек не технический и не обладающей достаточной квалификации именно по части управления проектом, скорее это продажник, который отвечает за общение с клиентом;
- тех-лиды на проектах фокусируются на решении инфраструктурных и технических проблем, не обладают достаточными полномочиями и квалификации для решения управленческих вопросов;
- поскольку проекты краткосрочные, сами команды не имеют достаточно времени чтобы запустились некие процессы самоорганизации внутри команд;
- периодические смены составов команд;
- быстрый рост компании, не уделяется достаточно внимания процессам внутри команд.

Сложившиеся процессы работы клиентских команд сложились сами по себе, при этом они в целом отражают суть бизнес цели - выпуск клиентского приложения в кратчайшие сроки. Но также они говорят о том, что упускается видение стратегических целей, т.к. в сложившихся условиях бизнес цель (по мнению автора этих строк) - создание системы для эффективного производства клиентских приложений. Для достижения этого необходимо выделить концепции, на основании которых уже должна формироваться методология работы команд, на мой взгляд это должны быть: Стандартизация, Модулизация, Автоматизация. Стандартизация -

использование единых подходов и правил для команд работающих над внутренними и клиентскими приложениями. С точки зрения Стандартизации, подходы и правила должны касаться не только инструментов, но и процессов, это позволяет производить одинаковый продукт одинаковыми способами, быстро переключать контекст в случае кросс-командного взаимодействия и использовать опыт решения проблем. Модулизация - по возможности деление функционала на модули для упрощения поддержки и версионирования. Автоматизация - вся работа, которую необходимо выполнять на разных проектах должна автоматизироваться.

Внутренних проектов несколько, в данной работе будет рассмотрена только мобильная разработка. Отдельная команда для работы была выделена только полгода назад и процессы еще не до конца установлены. На данный момент можно говорить о концепции быстрой разработки приложений (RAD). Это не формализовано и не является принятой стратегией, просто фактически существующие процессы более всего соответствуют этому подходу. Есть несколько причин, почему сложилась такая практика:

- на данный момент отсутствует руководитель проекта, целью которого бы стояло долгосрочное развитие проекта как продукта;
- команда состоит практически из разработчиков;
- ставятся цели по реализации нового функционала, который позволит продавать продукт;
- несоответствие прогресса по платформам (iOS и Android);
- большой список задач;
- нет системы, которая бы определяла этапы работ.

Все причины будут рассмотрены по отдельности с предложениями по улучшению процесса. На основании этого будет сделан вывод о том, является ли RAD подходящим подходом и если нет предложена наиболее подходящая методология.

Первый пункт - отсутствие руководителя. Формально на проекте есть руководитель, обладающий хорошими знаниями существующих продуктов, проводящий митинги и следящий за процессом. За процессом следят и другие руководители организации, также регулярно генерирующие идеи, предложения, участвующие в митингах по развитию проекта. Есть техлид,

который отвечает за приоритезацию задач, определение общего направления технического развития проекта. Если абстрактно посмотреть на эту ситуацию, то проблема заключается в том, что по сути есть потребители продукта, которые предъявляют к нему требования (маркетинг, руководители клиентских проектов), есть сам продукт и команда которая его разрабатывает, но нет в этой системе объекта, который отвечает за организацию взаимодействия между клиентом и производителем. Решение должно заключаться в определении лица (лиц) ответственного (и наделенного достаточными полномочиями для этого) за сбор требований их анализ/оценку, организацию взаимодействия с заинтересованными лицами (маркетинг, руководители клиентских проектов) и организацию процессов, позволяющих решать поставленные задачи и обеспечивающих долгосрочное развитие проекта как продукта.

Команда состоящая только из разработчиков способна выполнять задачи и улучшать продукт. При некоторых условиях это даже хорошо - отсутствие бюрократии, более быстрое принятие решений на уровне их полномочий - возможность сконцентрироваться на написании кода. Но задачи стоящие перед разработчиками не позволяют видеть картину целиком. Также в их полномочия не входят определение приоритетности задач и решение вопросов не связанных с разработкой.

В основном задачи на проекте представляют собой добавление новых фич. Однако для успешности долгоиграющего проекта этого не достаточно и этому есть много причин. Во-первых не всегда изначальная архитектура соответствует текущим требованиям. Во-вторых иногда требуется рефакторинг существующего кода, который изначально писался в других реалиях и под иные требования. В-третьих необходимо поддерживать мотивацию команды, а выполнение бизнес целей зачастую идет в разрез с этим. Опять же тут есть разные пути решения, например все это может быть отдано под контроль команды, но тогда есть вероятность того, что баланс изменится в обратную сторону и это пойдет в ущерб бизнес целям. Все это несколько выходит за рамки собственно каких-то конкретных подходов к управлению, и является общим вопросом философии управления. Тем не менее, очень важно это учитывать и процессы должны строиться таким образом, чтобы этому уделялось внимание на постоянной основе. Вполне естественно, что для подавляющего большинства руководителей проектов

рефакторинг и реструктуризация кода являются абсолютно бесполезными и ненужными вещами, поскольку они напрямую не добавляют продукту ценности, а скорее предотвращают задержки в будущем. С точки зрения разработчиков (надо признать что не всех) - это не является бесполезной работой, поскольку непосредственно влияет на качество их жизни. Наведение порядка в проекте является таким же важным фактором как и развитие. Игнорирование приводит к накоплению технического долга, замедляет работу и делает ее некомфортной, посредством этого снижает мотивацию и в итоге приводит к снижению производительности, а это уже является причиной задержек в реализации нового функционала, что напрямую влияет на бизнес цели. Поддержку существующего кода необходимо рассматривать не как бесполезные издержки, которые напрямую не влияют на ценность продукта, а как инвестицию времени в продукт, позволяющую в будущем предотвратить задержки и избежать потерь. Иногда разработчики способны видеть эти преимущества лучше руководителей, но поскольку в большинстве своем разработчики интроверты, им бывает сложно донести это до руководителей. На практике часто случается что люди увольняются по этим причинам - работать на проекте становится тяжело и неприятно и если отсутствует перспектива заниматься интересными задачами, проще уйти на другой проект. Ничего хорошего это не несет - уходя люди уносят знания и опыт, которые не были реализованы. Таким образом для долгосрочных проектов бизнес цель это не только эволюция с точки зрения повышения ценности для пользователя, но рост ценности как легко расширяемой и поддерживаемой системы. В идеале это не должно быть вынужденной мерой и не должно выполняться только по запросу со стороны команды разработки. Это должно быть естественной частью работы.

Несоответствие прогресса и в целом состояния проектов для разных платформ создает неудобства в версионировании. Тестирование в такой ситуации становится более трудозатратным. С точки зрения маркетинга это тоже неудобство - приходится учитывать это при презентации решения клиентам. По этой ситуации необходимо проводить отдельный анализ, учитывать это при оценке, приоритезации и планировании работ. Желательно синхронизировать задачи по разным платформам это упрощает коммуникацию внутри команды, позволяет снизить потери на переключении контекста разработчиков, тестеров и руководителя проекта. Методологии,

которые используют инкрементальный и итеративный подход позволяют решить это за счет определения плана на какой-то определенный отрезок времени, можно планировать задачи так, чтобы работа команд была синхронизирована по- максимуму по времени.

Большой перечень задач как правило является проблемой, поскольку им сложно управлять. В скраме есть понятие груминга - регулярный пересмотр бэклога (всего списка задач, которые еще не приняты к исполнению). Чем длиннее список задач, тем больше времени необходимо тратить на его проверку. Проверка бэклога должна включать в себя - приоритезацию, оценку времени необходимого на выполнение задачи, оценку необходимости выполнения задачи. Задачи, которые находятся в бэклоге более определенного периода времени являются кандидатами на удаление, так как в таком случае получается что ее ценность слишком низка, чтобы попасть в спринт и не стоит тратить время даже на ее пересмотр.

Отсутствие разделения списка на этапы затрудняет версионирование, синхронизацию работ по разным платформам и в целом не способствует эффективному планированию работы на будущее. Вместе с большим списком задач отсутствие поэтапного планирования может сказаться на мотивации. Человеческий мозг устроен таким образом, что мы получаем удовлетворение от достижения целей. Работа над нескончаемым списком задач превращает процесс в рутину. Завершение самой задачи уже не является достижением, поскольку за ней сразу следует другая. Поэтапная организация работы позволила бы во снизить монотонность работы, создать ощущение прогресса, а также проводить анализ полученного опыта и более эффективно планировать последующие этапы.

Рассмотрим методологии приведенные в первой части работы и попробуем выделить наиболее подходящие для внутреннего проекта. Водопадная модель видится наименее подходящей, поскольку: проект долгосрочный и требования нечетко определены - достаточно сложно этапировать работы с предварительным планированием, для осуществления планирования придется тратить очень много ресурсов и времени, слишком сложно организовать и формализовать взаимоотношения с клиентскими проектами. Как вариант возможно было бы рассмотреть что-то из модифицированных водопадных моделей, но опять же полностью побороть негибкость невозможно. Следующая модель - RAD, наиболее близкая к тому,

что на данный момент происходит на проекте. Во-первых необходимо отметить что она работает, но при этом есть ряд проблем, которые были рассмотрены выше и которые сложно решить используя эту модель. Следующие две модели лучше всего рассмотреть вместе, это итеративно-инкрементная модель и Scrum. Обе эти методологии предоставляют возможности для решения проблем описанных выше. В основе самих процессов лежит один и тот же базовый принцип - цикличность, но один подход на первый план ставит сам процесс развития продукта, другой описывает в первую очередь процессы внутри команды. Рассмотрим как эти два подхода могут решить существующие проблемы.

Проблему отсутствия руководителя сама методология не решит, это должно решаться другими путями, но Scrum по крайней мере дает понятие лица (в виде Product Owner - владельца продукта), которое должно отвечать за взаимодействие с клиентом (Stakeholders в Scrum).

С точки зрения Scrum кроме Product Owner больше не нужно никаких лиц ответственных за развитие проекта, поэтому нет никакой необходимости реорганизации существующей команды. Итеративно-инкрементная модель об этом ничего не говорит, но как сказано выше это проблема несколько выходит за рамки данного исследования.

Благодаря итеративности оба подхода могут помочь решить проблему, когда баланс развития продукта перекошен в сторону решения только бизнес целей. На этапе оценки и планирования (итеративно-инкрементная модель) и ретроспективы (Scrum) у команды есть возможность озвучить имеющиеся проблемы, проанализировать факторы которые замедляют разработку и изучить источники проблем. Далее это можно учитывать при планировании следующего цикла (спринта). Возможно внести какие-то задачи по улучшению проекта.

Итеративность обоих подходов, а также короткие циклы, позволяют наиболее эффективно синхронизировать работу разных членов команд. Анализ предыдущих итераций позволяет понять где необходимо усиление команды, и возможно изменения состава команд, привлечение новых людей для достижения соответствия прогресса. В случае когда команде приходится работать над разными задачами под разные платформы благодаря итерациям можно свести к минимуму дисбаланс и организовать поддержку внутри команды.

Scrum имеет достаточно четкие правила относительно списка задач (бэклог) и предоставляет рекомендации по работе с ним. Итеративно-инкрементная модель это не регулирует, но позволяет на этапе планирования определять то что будет реализовано в текущей итерации, что также несколько упрощает понимание прогресса.

Этапирование работ является основой обеих методологий. Благодаря этому можно реализовать простое и понятное версионирование продукта. Опять же благодаря инкрементам (спринтам) команды ощущают прогресс, клиенты также получают релизы регулярно и в некоем законченном виде. И самое главное каждый этап начинается и заканчивается планированием и анализом.

Какие выводы мы получили. Обе методологии являются хорошими инструментами которые позволят решить имеющиеся трудности. Scrum будет более предпочтительным решением, в случае если помимо решения производственных проблем, методология должна еще и решать вопросы организации команды и обеспечения позитивной рабочей атмосферы. В случае отсутствия квалифицированного руководителя с сильными софт-скиллами это будет более предпочтительным выбором и позволит сформировать более самостоятельную (в долгосрочной перспективе) команду.

ГЛАВА 3
ОСНОВНЫЕ НАПРАВЛЕНИЯ ПОВЫШЕНИЯ ЭФФЕКТИВНОСТИ УПРАВЛЕНИЯ ПРОЕКТАМИ ПО ПРОДУКТОВОЙ РАЗРАБОТКЕ МОБИЛЬНЫХ ПРИЛОЖЕНИЙ

3.1. Проблемы повышения эффективности управления проектами по продуктовой разработке мобильных приложений

На основании информации изложенной в первой и второй главах работы можно сделать ряд выводов о проблемах повышения эффективности управления проектами:

1. проекты по разработке мобильных приложений сильно отличаются друг от друга по своим характеристикам;
2. выбор методологии, а тем более ее адаптация или улучшение уже действующего подхода на проекте - сложная задача требующая понимания проекта и эмпирических знаний;
3. выбор правильного инструментария, соответствующего требованиям используемой методологии;
4. изменчивость долгосрочных проектов, что может требовать смены методологии полностью, либо адаптация части процессов;
5. модификация;
6. автоматизация бизнес процессов.

Рассмотрим по пунктам изложенные проблемы.

Разнохарактерность. Наиболее предпочтительной является ситуация, когда для разных проектов можно использовать одну и ту же методологию, но у универсальности как правило есть обратная сторона - универсальные решения могут все, но не решают конкретные задачи хорошо. Стандартизировать

Выбор методологии. Любой проект должен начинаться с анализа. До момента начала реализации должно быть четкое понимания характеристик

проекта и его особенностей. В данном случае речь не идет о сборе всех бизнес требований и составлении технического задания, а о таких характеристиках как оценка примерных сроков реализации проекта, будет ли проект долгосрочным, требуется ли поддержка, требуется ли исследовательская работа по реализации проекта (есть ли какой-то функционал для реализации которого не существует типовых решений, примером может быть проект с использованием искусственного интеллекта), критичность к устойчивости системы, неопределенность бизнес требований и прочие факторы которые увеличивают риски. На основании всей этой информации можно делать выводы. Если проекты краткосрочные и бизнес-требования ясны, то вполне можно обойтись каскадной моделью. При очень высоких требованиях к устойчивости и надежности системы стоит рассмотреть TDD подход (разработка через тестирование), либо использовать одну из итеративных методологий с внедрением практик TDD. Проекты для которых требуется максимально сократить сроки разработки с возможностью привлечение высококвалифицированных и мотивированных специалистов могут использовать RAD. Скрам будет хорошим выбором для долгосрочных проектов, которые требуют стабильно высокого качества продукта, быстрого реагирования на изменяющиеся требования и формирования самоорганизованной команды, которая требует минимума контроля с административной точки зрения.

Инструментарий. Подбор правильных инструментов позволяет максимально упростить процессы в команде. Есть решения, которые реализуют готовый набор функционала для конкретных методологий. Например JIRA имеет готовые решения для следующих подходов относящихся к Agile: Скрам, Канбан, смешанные методологии, масштабируемый Agile. Для каскадной модели вполне может подойти Trello или Pivotal Tracker как максимально простые для освоения и использования решения. Использование наиболее простых решений, соответствующих требованиям позволяет упростить процесс и сократить расходы.

Изменчивость. Необходимость изменения методологии, отдельных процессов или инструментов является нормальной ситуацией. Могут меняться требования, состав команд и другие внешние и внутренние условия. Отсутствие адаптации либо снизит производительность, либо приведет к другим негативным последствиям для проекта. Итеративность присущая

некоторым методологиям должна заключаться не только в анализе того “что” мы делаем, но и “как” мы делаем.

Модификация. Использование стандартных подходов (например Скрам в чистом виде) несет в себя много преимуществ. Стандартные отработанные решения проще внедрять. При смене команд человеку проще адаптироваться. Прозрачнее с точки зрения управления. Заказчикам (если им приходится работать с несколькими командами) также легче взаимодействовать при одинаковых правилах. Однако в некоторых ситуациях существующие методологии могут быть модифицированы для того чтобы лучше соответствовать производственным процессам проекта. Первый пункт ценностей Agile манифеста гласит: *люди и взаимодействие важнее процессов и инструментов.* Этот простой принцип является наиболее важным с точки зрения повышения эффективности подходов к управлению проектами. Изучение практик использующихся в других методологиях, постоянный анализ проблемных мест и исследование способов улучшения имеющихся подходов, а также открытость к изменениям - путь к нахождению наиболее оптимальных стратегий.

Автоматизация. В контексте подбора инструментария необходимо упомянуть и вопрос автоматизации. Одним из преимуществ использования методологий которые следуют итеративным или инкрементальным принципам является то, что регулярно повторяющиеся циклы (причем чем короче цикл, тем более явно это прослеживается) позволяют выделять работу, которая повторяется на каждом из циклов. Самым важным при решении какой-то проблемы является осознание - циклы позволяют определить работу, которая регулярно повторяется и автоматизировать ее. Причем это касается не только автоматизации производственных процессов (связанных с кодом или другими техническими аспектами), но и автоматизировать часть управленческой работы - изменение статусов задач, сбор управленческой аналитики и подготовка отчетов. Простым примером может быть использование инструментов для отслеживания статистики стабильности приложений. Вместо регулярной ручной проверки, интеграции сервисов позволяют отслеживать это автоматически и рассылать уведомления в случае превышения пороговых значений, либо даже автоматическое создание задач. Такое простое решение экономит время команды и ускоряет реагирование на возникающие проблемы.

3.2. Концептуальные подходы к формированию продуктовой стратегии разработки мобильных приложений

Компаниям осуществляющим разработку программного обеспечения необходимо учитывать особенности производимого продукта. В сфере разработки мобильных приложений это даже сильнее прослеживается, поскольку эта сфера еще более динамична. Специфика программного обеспечения как продукта заключается в следующем:

- программное обеспечение это сложный товар имеющий научно-технологическую природу;
- отсутствие физического носителя;
- отсутствие географической привязки к месту производства и потребления (данный случай может иметь ограничения, например законодательные, либо локализация);
- в процессе потребления не устаревают (в физическом смысле) и не расходуются;
- очень быстро устаревают.

С точки зрения процесса потребления мобильных приложений как продукта можно выделить следующие черты. Пользователи могут использовать приложения по-разному. Использование предполагает наличия технической поддержки, это также является уникальном отличием программных продуктов - они могут улучшаться и изменяться уже в процессе потребления.

Несмотря на возможность защиты от копирования, лицензирования и правовой защиты программных продуктов они могут быть легко скопированы, при этом защитить права - довольно сложная задача.

Еще одна интересная черта именно продуктовой разработки в том, что нет жесткой прямой связи между финансированием и скоростью производства продукта [23].

Особенности продуктовой разработки. Прежде всего следует отметить, что высокотехнологичные компании, производящие ПО, характеризуются рядом особенностей продуктовой разработки, маркетинга и продаж, свойственных разве что фармацевтике и биотехнологии: большие

ин-вестиции в НИОКР при ничтожно малой себестоимости конечных продуктов.

В любом высокотехнологичном производстве существуют три группы рисков: технологические (связанные со сложностью и многообразием технологий), финансовые (связанные с необходимостью крупных инвестиций при отсутствии гарантии создания коммерчески успешных продуктов), временные (связанные с длительностью сроков разработки продуктов).

Те же риски имеются у компаний, производящих программное обеспечение, за исключением следующих отличий:

1. рынок информационных технологий не является сформировавшимся (как, например, рынок фармацевтики и биотехнологий) и вряд ли когда-нибудь сформируется, поскольку основным двигателем являются новые технологии и направления, которых ранее не существовало, поэтому практически каждый новый продукт создает спрос на него, а не наоборот;
2. программные продукты не самодостаточны, то есть их использование требует вовлечения определенных аппаратных средств, а также других программных продуктов;
3. как правило, текущие потребности рынка не совпадают со временем выпуска продуктов или находятся на ранней стадии осознания и формирования этих потребностей, поэтому для создания продукта необходима сильная прогнозная составляющая, то есть способность предсказывать тренды развития технологий или даже общества на 1–3 года вперед;
4. разработка ПО требует задействования в производственном процессе специфических человеческих ресурсов, а именно высокоинтеллектуальных ресурсов, обладающих уникальными, трудно тиражируемыми навы-ками и знаниями, а следовательно, редких. В данный момент на рынке труда спрос на них существенно превышает предложение.

Каждый новый успешный продукт в сфере информационных технологий является результатом новых идей и концепций, предсказуемость появления которых крайне низка. В этой связи представляется необходимой перегруппировка существующих рисков посредством их объединения следующим образом: маркетинговые риски; технологические риски; риски,

связанные с человеческим фактором. Все они приводят к возникновению финансовых, временных и репутационных рисков.

Маркетинговые риски:

- неправильная оценка потребностей рынка;
- ошибка в ценообразовании;
- недооценка временного лага в привыкании рынка к новой продукции;
- неправильный выбор времени запуска продаж продукта;
- недостаточная степень адаптированности продукта под нужды региональной целевой ауди-тории (локализации);
- замещение продукта субститутами из других отраслей.

Технологические риски:

- неправильная архитектура продукта;
- невозможность технологической реализации требований;
- несоответствие полученного продукта технологическим требованиям;
- несовместимость продукта с основным ПО потребителей;
- критически большое количество ошибок в коде продукта;
- не масштабируемость продукта;
- выпуск слишком «тяжелого» продукта, потре-бляющего чрезмерно много ресурсов;
- недостаточная производительность продукта.

Риски, связанные с человеческим фактором:

- неправильная оценка сроков разработки проекта;
- неправильная оценка требуемых ресурсов.

Финансово-временные риски:

- время и стоимость разработки превышает планируемые нормативы;
- ошибка с количеством циклов разработки и тестирования;
- недооценка аппаратных или программных средств разработки;
- невозможность выпуска продукта в объявленные на рынке сроки.

Выводы следующие из изложенного:

1. отрасль информационных технологий является крайне специфичной по сравнению с традиционными отраслями производства;
2. экономические законы и подходы к стратегическому планированию отработанные десятилетиями и веками не работают в этой сфере;
3. необходимо проведение большого количества работ по исследованию и разработке (R&D - research and development);
4. деятельность в этой сфере довольно рискованная и с трудом поддается планированию, при этом процент успешных проектов среди стартапов очень низок, цифры колеблются от 10% и ниже - все зависит от методики оценки [24].

Таким образом приведенные выше риски и выводы об особенностях продуктовой разработки программного обеспечения требуют учета в продуктовой стратегии, увеличивая прогнозируемый бюджет и прогнозные сроки продуктовых проектов. Разработка подходов к формированию продуктовой стратегии должна включать ряд мер, которые можно разделить на: управленческие, технические и маркетинговые.

Управленческие:

- использование подходящих методологий управления проектами;
- внедрение систем мотивации, а также создании комфортной атмосферы;
- использование таких практик как теория решения изобретательских задач.

Технические:

- использование грамотных архитектурных решений (дробление проекта на несколько подпроектов, проектирование легко расширяемых решений);
- автоматизация всех процессов (управленческих, технологических);
- использование высоких стандартов качества кода, автоматическое тестирование;
- тестирование продукта, включая A/B тестирование начинающееся с ранних фаз.

Маркетинговые:

- продвижение продукта с ранних стадий;
- поиск партнеров и клиентов;
- рекламные кампании, особенно привязанные к релизам.

Продуктовая стратегия разработки приложений является комплексным мероприятием. Она должна включать в себя подходы на всех уровнях: управленческом, техническом, маркетинговым. Продуктовая разработка - высоко рискованное мероприятие до момента захвата значительной доли рынка, но даже после этого остаются риски ее потери. Для разработки стратегии применим цикл PDSA (Plan-Do-Study-Act), продуктом в данном случае может быть дорожная карта проекта - стратегический план компании.

ЗАКЛЮЧЕНИЕ

Институт управления проектами (Project Management Institute) - всемирная некоммерческая организация по управлению проектами до 3 версии (2004 год) Свода знаний по управлению проектами (Project Management Body of Knowledge, PMBoK) предлагал использовать “каскадную модель” как основную методику управления. Только в 2009 году с выходом 4-й версии PMBOK предлагалось использовать гибридный вариант методологии управления проектами, сочетающий в себе как плюсы от каскадной модели, так и достижения итеративных методологов [10]. Практики гибкой модели были включены в руководство только в 2017 году с выходом 6-й версии книги. Все эти факты говорят о том, что водопадная модель широко используется, хорошо стандартизируется и даже является предпочтительной в определенных случаях. Однако, исследовав историю появления и развития других моделей (итеративно-инкрементная, Agile) можно отметить:

- они существуют давно;
- использовались для реализации очень крупных и важных проектов;
- эксплуатантами (и создателями) являются очень серьезные организации, такие как NASA, IBM, Microsoft и прочие.

Хочется остановиться на этом моменте немного подробнее. Почему же все таки итеративно-инкрементная модель, как и Agile так поздно были признаны и включены в свое руководство Институтом управления проектами? Основным фактором тут на мой взгляд является стандартизация. Чем более формализован подход и этапы, тем проще предъявлять требования и стандартизировать знания/умения сотрудников. Если на разных проектах используются одни и те же стандарты, процессы, документация, тем проще заменить одних людей другими, при этом это окажет минимальное воздействие на проект (при одинаковой квалификации).

Следующий фактор отличия - происходит некоторое смещение важности со строго вертикального типа управления проектом к горизонтальному. Практики гибкой методологии в принципе не касаются вопроса руководства. Scrum описывает команду как самодостаточную сущность и не предполагает какого-нибудь руководства извне. Это оставляет не у дел

сертифицированных руководителей, но приводит к появлению таких практик как Agile release train.

Наличие достаточно квалифицированного и мотивированного персонала способного обеспечить работу эволюционных и гибких подходов. IID и Agile предполагает постоянный анализ происходящего, практики Agile предполагают что это должно производиться командой, которая работает над проектом, что налагает еще более высокие требования к профессионализму сотрудников.

Развитие инструментов. Возьмем для примера системы управления базами данных. Первая бесплатная база данных MySQL появилась только в 1995 году, до этого были только коммерческие решения. На данный момент существует множество решений, которые соответствуют международным стандартам и позволяют решать широкий круг задач. Это же касается и других инструментов разработки. Их наличие значительно упрощает и ускоряет процесс разработки, что с одной стороны позволяет маленьким командам решать все более сложные задачи, а с другой снижает требования к технической квалификации, позволяя сотрудникам сконцентрироваться на решении более высокоуровневых задачах и брать на себя больше функций и ответственности, которые обычно возложены на менеджеров.

Также эти модели сложнее описать научными методами, описать процессы. Необходимо было сначала подвести научную методологию для этих моделей. Как выразился Филипп Круштен, профессор в области разработки программного обеспечения Университета Британской Колумбии, Ванкувер, Канада [11]:

> *“The agile movement is in some ways a bit like a teenager: very self-conscious, checking constantly its appearance in a mirror, accepting few criticisms, only interested in being with its peers, rejecting en bloc all wisdom from the past, just because it is from the past, adopting fads and new jargon, at times cocky and arrogant. But I have no doubts that it will mature further, become more open to the outside world, more reflective, and therefore, more effective.” — Philippe Kruchten [12]*

> *“Agile направление в некотором роде как подросток: очень застенчивый, постоянно поглядывающий в зеркало, не*

принимающий критики, не заинтересованный в общении со сверстниками, отвергающий мудрость опыта, только лишь потому что это из прошлого, с причудами и своим жаргоном, временами дерзкий и высокомерный. Но я не сомневаюсь, что он будет взрослеть, становиться более открытым к внешнему миру, рефлексивным и таким образом более эффективным"

Методологии, относящиеся к группе Agile, уже прочно завоевали место в сфере разработки программного обеспечения и не только. Agile подход уже принят авторитетными организациями в сфере управления проектами (Project Management Body of Knowledge, PMBoK) [19]. Некоторые исследователи сравнивают PMI ACP (Project Management Institute Agile Certified Practitioner - сертифицированный управленец использующий практики Agile, выдается Институтом Управления Проектами) и CSM (Certified Scrum Master - сертифицированный скрам мастер, выдается организацией Scrum Alliance) [20]. Интересен тот факт, что Скрам фреймворк не противоречит PMBoK - еще в четвертом издании книги был введен итеративный подход (plan-do-check-act), таким образом использующий Скрам руководитель проекта фактически продолжает придерживаться рекомендаций PMI [21].

Неоспоримым преимуществом гибких методологий является простота понимания. Камнем преткновения становится сложность применения на практике. В первую очередь это относится к Скраму, но по сути свойственно практически любой методологии семейства Agile. Причина заключается в том, что эффективность достигается за счет изменения подходов к работе, нахождении узких мест. Это требует изменения привычных паттернов поведения, искоренения неэффективных практик. В большинстве случаев именно люди и привычки, являются причиной проблем, сложность в том, что менять укоренившиеся привычки тяжело и не все готовы на это.

Применение методологий рассмотренных в работе выходит за рамки сферы информационных технологий. Каскадная модель используется повсеместно. Но и Agile подходы находят применение. Данным в целом по семейству Agile нет, но есть данные по Скраму. Согласно отчету State of

Scrum 2017-2018, Скрам используется для небольших рабочих групп в следующих сферах:

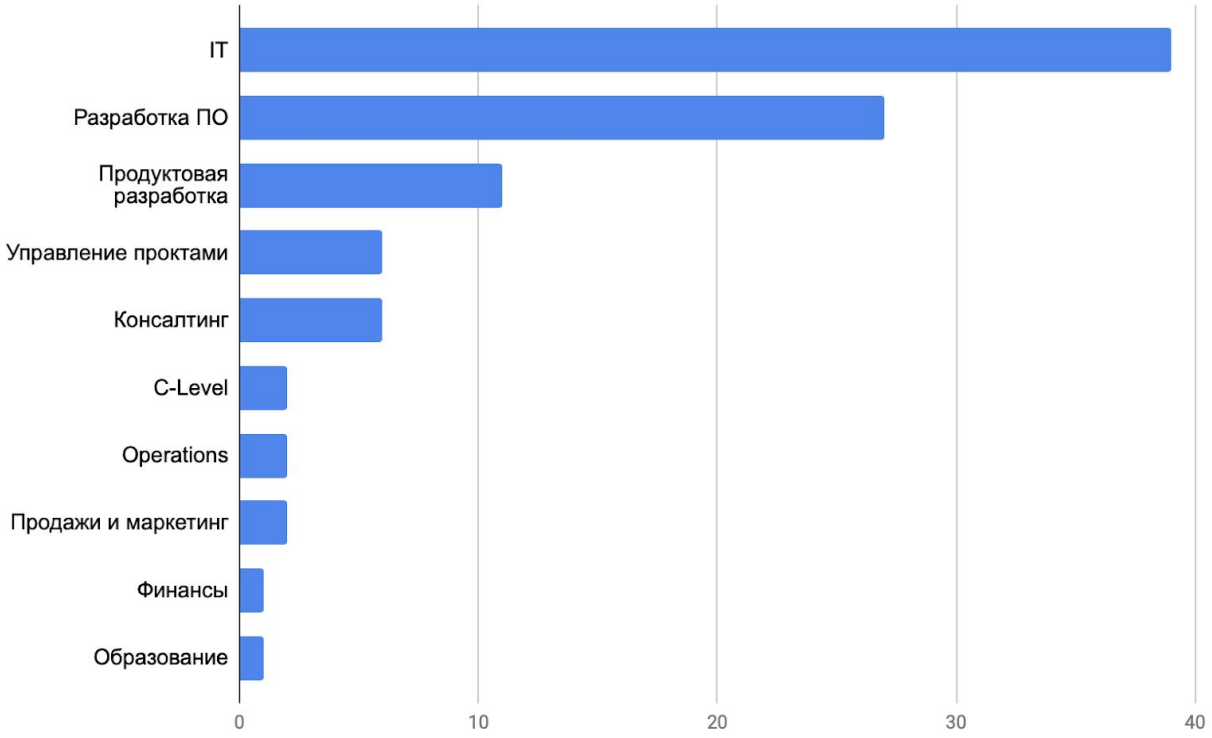

Рисунок 4.1 - Распределение респондентов использующих Скрам по сферам деятельности.

Примечание - Источник [22].

Очевидно что IT сфера доминирует, однако около трети респондентов применяющих Скрам в своей работе относятся к другим сферам. С учетом роста популярности Agile модели и профессиональной миграции между сферами, можно прогнозировать большее проникновение гибких методологий в другие сферы. Наибольший потенциал имеет сфера образования (1% респондентов). Польза изучения заключается в освоении новых подходов к управлению образовательными проектами, получении полезного практического опыта в части управления проектами и осмыслении базовых принципов эффективного развития.

В работе были рассмотрены следующие подходы к управлению проектами: каскадная модель, модифицированные каскадные модели, итеративно-инкрементная разработка, быстрая разработка приложений, Скрам, Agile. Эта последовательность показывает основные этапы развития

подходов к управлению проектами по разработке программного обеспечения и отражает их эволюцию.

На примере реально действующих проектов были рассмотрены некоторые проблемы управления проектами по разработке мобильных приложений и способы их преодоления, предложены некоторые рекомендации по возможности применения различных методологий, а также рассмотрены наиболее популярные инструменты для организации процесса управления проектами и взаимодействия команд.

СПИСОК ИСПОЛЬЗОВАННОЙ ЛИТЕРАТУРЫ

1. A review of enterprise agility: Concepts, frameworks, and attributes Bohdana Sherehiy, Waldemar Karwowski, John K. Layer [Электронный ресурс]. - Режим доступа: https://www.researchgate.net/publication/223752105_A_Review_of_Enterprise_Agility_Concepts_Frameworks_and_Attributes - Дата доступа: 01.04.2019
2. United States, Navy Mathematical Computing Advisory Panel (29 June 1956), Symposium on advanced programming methods for digital computers, Washington, D.C., Office of Naval Research, Dept. of the Navy, 1956
3. Bell, Thomas E., and T. A. Thayer. Software requirements: Are they really a problem? Proceedings of the 2nd international conference on Software engineering, IEEE Computer Society Press, 1976
4. Military Standard Defense System Software Development [Электронный ресурс]. - Режим доступа: https://www.product-lifecycle-management.com/download/DOD-STD-2167A.pdf - Дата доступа: 01.042019
5. McConnell, Steve (1996). Rapid Development: Taming Wild Software Schedules, Microsoft Press, 1996
6. Larman, Craig (June 2003). "Iterative and Incremental Development: A Brief History" [Электронный ресурс]. - Режим доступа: http://www.craiglarman.com/wiki/downloads/misc/history-of-iterative-larman-and-basili-ieee-computer.pdf - Дата доступа: 01.05.2019
7. Statistical method from the viewpoint of quality, Dover, 1986 (reprint from 1939) [Электронный ресурс]. - Режим доступа: http://library.isical.ac.in:8080/jspui/bitstream/123456789/6845/1/Statistical%20method%20from%20the%20viewpoint%20of%20quality%20control.pdf - Дата доступа: 01.05.2019
8. Achieving Effective Acquisition of Information Technology in the Department of Defense [Электронный ресурс]. - Режим доступа:

https://books.google.by/books?id=WDRkAgAAQBAJ&pg=PT65&lpg=PT65&dq=Project+Mercury+iterations+day&source=bl&ots=0l07FQ_0Ao&sig=ACfU3U242oEGLuqk_WYHx-7t7DhaYvcvyg&hl=ru&sa=X&ved=2ahUKEwjOnIOVuo_iAhVBJFAKHUEDA0cQ6AEwEnoECAYQAQ - Дата доступа: 01.05.2019

9. Jeff Sutherland [Электронный ресурс]. - Режим доступа: https://en.wikipedia.org/wiki/Jeff_Sutherland - Дата доступа: 01.05.2019

10. Каскадная модель [Электронный ресурс]. - Режим доступа: https://ru.wikipedia.org/wiki/Каскадная_модель. Дата доступа: 10.05.2019

11. Philippe Kruchten [Электронный ресурс]. - Режим доступа: https://en.wikipedia.org/wiki/Philippe_Kruchten - Дата доступа: 10.05.2019

12. Agile's Teenage Crisis?, Kruchten Philippe, InfoQ, 2011

13. Manifesto for Agile Software Development, Kent Beck; James Grenning; Robert C. Martin; Mike Beedle; Jim Highsmith; Steve Mellor; Arie van Bennekum; Andrew Hunt; Ken Schwaber; Alistair Cockburn; Ron Jeffries; Jeff Sutherland; Ward Cunningham; Jon Kern; Dave Thomas; Martin Fowler; Brian Marick, Agile Alliance, 2010. [Электронный ресурс]. - Режим доступа: http://agilemanifesto.org/. - Дата доступа: 10.05.2019

14. Business Object Design and Implementation Workshop [Электронный ресурс]. - Режим доступа: http://jeffsutherland.com/oopsla/oo95wrkf.html - Дата доступа: 10.05.2019

15. Руководство по Скраму на русском языке, Кен Швабер, Джефф Сазерленд [Электронный ресурс]. - Режим доступа: https://scrumguides.org/docs/scrumguide/v2017/2017-Scrum-Guide-Russian.pdf - Дата доступа: 10.05.2019

16. Why Software Fails, Robert N. Charette [Электронный ресурс]. - Режим доступа: https://spectrum.ieee.org/computing/software/why-software-fails/3. - Дата доступа: 10.05.2019

17. Principles of Effective Story Writing: The Pivotal Labs Way [Электронный ресурс]. - Режим доступа: https://www.pivotaltracker.com/blog/principles-of-effective-story-writing-the-pivotal-labs-way - Дата доступа: 04.06.2019

18. Agile tools for software teams [Электронный ресурс]. - Режим доступа: https://www.atlassian.com/software/jira/agile - Дата доступа: 02.06.2019

19. A guide to the project management body of knowledge (PMBOK®Guide) (4th ed.). Newtown Square, PA: Project Management Institute, 2008

20. PMI ACP vs CSM Comparison: Which One Is Better? [Электронный ресурс]. - Режим доступа: https://blog.masterofproject.com/pmi-acp-vs-csm/ - Дата доступа: 02.06.2019

21. Agile project management with Scrum. Paper presented at PMI® Global Congress 2011—North America, Dallas, TX. Newtown Square, PA: Project Management Institute, 2011 [Электронный ресурс]. - Режим доступа: https://www.pmi.org/learning/library/agile-project-management-scrum-6269 - Дата доступа: 02.06.2019

22. State of scrum [Электронный ресурс]. - Режим доступа: https://www.scrumalliance.org/ScrumRedesignDEVSite/media/ScrumAllianceMedia/Files%20and%20PDFs/State%20of%20Scrum/2017-SoSR-Final-Version-(Pages).pdf - Дата доступа: 02.06.2019

23. Соловьев В. И. Стратегия и тактика конкуренции на рынке программного обеспечения: Опыт экономико-математического моделирования: монография. М.: Вега-Инфо, 2010. С. 24–28.

24. Почему «взлетает» только 1% стартапов — и это нормально [Электронный ресурс]. - Режим доступа: https://www.forbes.ru/tehnologii/339113-pochemu-vzletaet-tolko-1-startapov-i-eto-normalno - Дата доступа: 03.06.2019

25. Production of Large Computer Programs [Электронный ресурс]. - Режим доступа:

https://sunset.usc.edu/csse/TECHRPTS/1983/usccse83-501/usccse83-501.pdf - Дата доступа: 05.04.2019

26. What is the ideal size of a Scrum team? [Электронный ресурс]. - Режим доступа: https://www.quora.com/What-is-the-ideal-size-of-a-Scrum-team - Дата доступа: 02.06.2019

27. McConnell, Steve. Code Complete, 2nd edition. Microsoft Press, 2004. - 914 с.

28. Michael Sahota. An Agile Adoption And Transformation Survival Guide. InfoQ, 2012. - 67 с.

29. Stephen R. Palmer, John M. Felsing. A Practical Guide to Feature-Driven Development. Pearson Education, 2001.

30. State of scrum [Электронный ресурс]. - Режим доступа: https://www.scrumalliance.org/ScrumRedesignDEVSite/media/ScrumAllianceMedia/Files%20and%20PDFs/State%20of%20Scrum/2017-SoSR-Final-Version-(Pages).pdf - Дата доступа: 02.06.2019

31. Agile - гибкий подход к управлению проектами [Электронный ресурс]. - Режим доступа: https://www.youtube.com/watch?v=lYp_OYscUDQ - Дата доступа: 05.05.2019

32. Scrum.org Introduces Scrum with Kanban Course, Enabling Greater Transparency Among Development Teams [Электронный ресурс]. - Режим доступа: https://www.businesswire.com/news/home/20180226005111/en/Scrum.org-Introduces-Scrum-Kanban-Enabling-Greater-Transparency - Дата доступа: 04.06.2019