УДК 004.424

СОВРЕМЕННОЕ СОСТОЯНИЕ И ПРОБЛЕМЫ РАЗРАБОТКИ ПРОГРАММНОГО ОБЕСПЕЧЕНИЯ С ИСПОЛЬЗОВАНИЕМ ГИБКИХ МЕТОДОЛОГИЙ (реферат)

Мухамадеева Рената Маратовна, магистр группы М21-904 «Бизнес-информатика в цифровой экономике», факультет очного обучения НИЯУ МИФИ

Брюхова Елена Михайловна, научный руководитель, аспирант кафедры №71 «Экономика и менеджмент в промышленности» НИЯУ МИФИ

Аннотация

Для преодоления вызовов, связанных с изменениями требований, которые существуют не только в сфере разработки программного обеспечения, но и в проектах разработки новых продуктов, практики и инструменты гибких методологий (Agile) имеют высокую ценность. Несмотря на рост популярности гибких методологий, в мире на сегодня отсутствует необходимое количество информации и специалистов, обладающих экспертизой эффективного использования Agile. В связи с этим возникает проблема определения подходов, которые позволят организациям максимизировать эффективность использования Agile.

КЛЮЧЕВЫЕ СЛОВА: программное обеспечение, водопадная модель, методология Agile.

CURRENT STATUS AND PROBLEMS OF SOFTWARE DEVELOPMENT USING FLEXIBLE METHODOLOGIES (abstract)

Mukhamadeeva Renata Maratovna, master of group M21-904 “Business informatics in the digital economy”, faculty of full-time education, National Research Nuclear University MEPhI

Bryukhova Elena Mikhailovna, scientific adviser, postgraduate student of the department №71 “Economics and management in industry” National Research Nuclear University MEPhI

Abstract

To overcome the challenges of changing requirements that exist not only in the field of software development, but also in new product development projects, Agile practices and tools are of great value. Despite the growing popularity of agile methodologies, the world today lacks the necessary amount of information and specialists with expertise in the effective use of Agile. In this regard, the problem arises of determining approaches that will allow organizations to maximize the effectiveness of using Agile.

KEYWORDS: software, waterfall model, Agile methodology.

Введение

В современном мире экономика нестабильна и рынок находится в состоянии постоянных изменений. В условиях глобальной конкуренции и общего повышения требований конечных потребителей к качеству товаров или услуг, организациям приходится регулярно вносить изменения в выпускаемую продукцию и оказываемые услуги.

В связи с этим большая часть традиционных методов проектного управления начинает устаревать, это связано с быстро меняющейся внешней средой, становится более

рациональным внедрять такие технологии, которые могут обеспечить гибкость решений при управлении проектами.

Несмотря на то, что на сегодняшний момент применение методологии Agile в большей степени практикуется именно в деятельности, связанной с информационными технологиями и смежными с ней отраслями, так как создание данной методологии изначально имело цель устранить вопросы и проблемы проектного управления именно в этой сфере, однако доля компаний, которые никак не связаны с информационными технологиями, но используют этот метод, постоянно растет.

Для преодоления вызовов, связанных с изменениями требований, которые существуют не только в сфере разработки программного обеспечения, но и в проектах разработки новых продуктов, практики и инструменты Agile имеют высокую ценность, так как данная методология изначально появилась как альтернатива традиционным подходам к разработке программного обеспечения, которые несли в себе достаточное количество слабых мест, имеющих большие проблемы для гибкой среды, изменяющихся требований ИТ-проектов.

Несмотря на рост популярности гибких методологий в мире отсутствует необходимое количество информации и специалистов, обладающих экспертизой эффективного использования Agile, вследствие чего компаниям зачастую не удается достигнуть поставленных целей при помощи гибкого управления. В связи этим возникает проблема определения подходов, которые позволят организациям максимизировать эффективность использования Agile. Для рассмотрения данной проблемы необходимо решить следующие задачи:

- Изучить проблемы использования традиционных методов, для решения которых были предложены методы Agile
- Изучить принципы и идеи Agile
- Изучить Agile-подходы к разработке программного обеспечения
- Определить практику к применению Agile
- Выполнить анализ использования Agile в России и мире
- Определить подход, который позволит максимизировать ценность Agile

1. История развития гибких методологий и обзор их предшественников

1.1. Общие сведения о гибких методологиях разработки

Большое количество исследований освещают гибкие методологии разработки как ответ на устаревшие традиционные методы. Однако идеи гибких методологий зарождались в 70-х годах прошлого века и ранее. Группы людей, находившие традиционные методы

неэффективными, предлагали альтернативные методы разработки, содержащие в себе идеи Agile, но на протяжении 30 лет данные идеи не воспринимались всерьез и компании не уходили от традиционных методов. В данной главе будет рассмотрена история развития Agile – мышления и таких аспектов Agile как реакция на изменения, вовлечение заказчиков и документирование программного обеспечения.

Гибкой методологию делают такие качества как адаптивность, итеративность и инкрементность и ориентированность на людей. Рассмотрим каждое из этих качеств.

1. Адаптивность

   Гибкие методологии приветствуют изменения, как в технологиях, так и в требованиях, и, более того, в самих методологиях. Адаптивный процесс- это процесс, который может дать контроль над непредсказуемостью. [1]

2. Итеративность и инкрементность

   Программное обеспечение разрабатывается в несколько итераций, каждая из которых проходит весь процесс разработки, от планирования до вывода в эксплуатацию. На каждой итерации часть системы разрабатывается, тестируется и дорабатывается, в то время как новая часть системы только начинает разрабатываться. В каждой итерации функциональность системы как правило должна улучшаться. Система развивается инкрементно, поскольку новые функции и подсистемы добавляются с каждой последующей итерацией. После каждой итерации версия системы предоставляется заказчику для получения обратной связи.

3. Ориентированность на людей

   «Люди важнее любого процесса» [2] В гибких методологиях люди – это основной двигатель успеха проекта. Роль процесса в рамках гибких методологий заключается в том, чтобы помочь команде разработчиков определить наилучший способ выполнения работы.

1.2. Традиционные методы разработки программного обеспечения

Несмотря на то, что итеративные и инкрементные подходы используются на протяжении многих лет, некоторые источники советуют использовать строго последовательный процесс разработки, известный как Водопад или Каскадная модель. Такая модель подразумевает, что переход к следующему этапу разработки возможен только после полного завершения предыдущего этапа [3]. Ученые выявили множество проблем, возникающих с применением данной методологии, среди которых недостаточная гибкость

подхода и следствия строгого управления разработкой в виде ущерба срокам, качеству и ресурсам. Взамен водопадной модели были предложены V-Model [4,5].

Перечисленные подходы были созданы для того, чтобы решить проблемы водопадной модели, однако они все еще оставались «жесткими» и привязанными к документации подходами. Фаулер называет эти подходы инженерными методологиями, которые идеально подходят для построения моста, но не для создания программного обеспечения, поскольку создание программного обеспечения — это другой вид деятельности, и для него нужен другой процесс. [1] По словам Хайсмита и Кокберна, Agile-методы были предложены с «точки зрения, отражающей внезапные изменения в бизнесе и технологиях». [6] Традиционные подходы не могли справиться с этим изменением, поскольку они предполагают, что можно предвидеть полный набор требований на ранней стадии жизненный цикл проекта. В реальности большинство изменений в требованиях к проекту и используемых технологиях происходят в течение всего жизненного цикла разработки ПО.

1.3. Зарождение принципов Agile

Большинство из идей Agile не такие новые, как кажется. Многие разработчики в конце прошлого века считали, что это наиболее успешный способ создания программного обеспечения. Однако к идеям Agile не относились серьезно, и поэтому, представление их как подхода к разработке программного обеспечения является новым. [7]

Считается, что Итеративная и инкрементальная разработка (Iterative and incremental development - IID) является сердцем любой Agile-методологии. Люди успешно использовали эти подходы в 70-х и 80-х годах прошлого века. Ларман и Басили нашли первые корни итеративной IID с 1950-х годов в NASA и IBM Federal Systems Division (FSD) [8.]. По их словам, проект NASA «Меркурий» в 1961-1963 гг. проводился с «короткими итерациями по полдня». Кроме того, была применена практика экстремального программирования, когда сначала планировались и писались тесты, и только затем был написан код для прохождения тестов. Также они использовали непрерывную интеграцию, поскольку каждая мини-итерация требует интеграции всего кода и прохождения тестов.

В 1970 году Уинстон Ройс, критиковавший Каскадную модель, рекомендовал «пять дополнительных функций, которые должны быть добавлены к традиционному подходу для устранения большинства рисков, возникающих при разработке» [9]. Эти шаги имели черты итеративной разработки.

Также другие ученые, критиковавшие Каскадную модель, предлагали альтернативные подходы. Например, своей статье "Stop the Life-cycle, I Want to Get off." [10] Глэдден предложил новый взгляд на процесс разработки и назвал его «нециклической голливудской моделью». По словам Глэддена, эта модель удовлетворяет трем положениям [10]:

1. Цели системы важнее, чем системные требования. Данное положение соответствует идее Agile, заключающейся в важности общего понимании системы, а не в наличии подробных требований, которые будут меняться в ходе проекта.
2. Физический объект передает больше информации, чем письменная спецификация. Это положение также отмечается как одна из ценности манифеста Agile: рабочее программное обеспечение важнее документации.
3. Определенные цели системы и демонстрации системы в совокупности приведут к успешному результату. Под успешным результатом стоит понимать продукт, который выполняет предназначенную функцию и удовлетворяет потребности клиента.

Глэдден считает, что большинство клиентов не имеют четкого представления о своих потребностях. Кроме того, он поднял проблему отсутствия и изменения требований.

Еще одно предложение было высказано Маккракеном и Джексоном в их статье «Lifecycle Concept Considered Harmful». Они предложили два сценария процессов развития системы [11]:

1. Прототипирование: авторы статьи предложили создать прототип на раннем этапе процесса разработки, согласно первым пользовательским требованиям. Серия прототипов или серия модификаций первого прототипа постепенно приведет к окончательному продукту. Именно такой и должна быть разработка по Agile: с короткими итерациями, каждая из которых улучшает систему. Кроме того, они рекомендовали поддерживать тесные отношения с пользователем (клиентом): «разработка идет шаг за шагом вместе с пользователем по мере накопления информации о собственной среде и потребностях пользователя».
2. Вторым предложением был процесс разработки системы, выполняемый в следующей последовательности: реализация, проектирование, спецификация, перепроектирование, повторное внедрение. Начинать с внедрения системы — это также идея современной итеративной разработки.

В 1985 году Том Гилб написал статью «Evolutionary Delivery versus the «waterfall model». В этой статье Гилб представляет метод EVO как альтернативу Водопадной модели,

которую он считал «нереалистичной и опасной для любого программного обеспечения» [12]. Гилб основал EVO на трех принципах:

1. Сделать продукт, соответствующий пользовательским требованиям
2. Снизить затраты за счет анализа и исправления проблем на каждом этапе жизненного цикла
3. Реагировать на изменения

Еще одна важная концепция в методе EVO - «Полный анализ, проектирование и тестирование на каждом этапе». Гилб подчеркнул, что его метод ориентирован на конечного пользователя: «Разработчику специально поручено своевременно и часто прислушиваться к обратной связи пользователей. Пользователь может принимать непосредственное участие в процессе разработки».

Очевидно, что многие концепции Гилба соответствуют принципам Agile, и не только частые поставки и короткие итерации, но также и роль пользователя в процессе разработки.

После EVO в 1988 году компания DuPont представила методологию под названием Rapid Iterative Production Prototyping (RIPP). Основная цель заключалась в создании прототипов, которые можно было бы регулярно представлять клиентам, чтобы гарантировать, что готовый продукт будет соответствовать пользовательским требованиям.

Джеймс Мартин расширил эту методологию до формализованной методологии, которая была переименована в Rapid Application Development (RAD). Жизненный цикл RAD состоит из четырех этапов: планирование требований, пользовательское проектирование, создание и внедрение. Отличительной особенностью RAD было детальное проектирование и разработка одной версии за другой. Каждая версия представлялась конечным пользователям для внесения исправлений. Кроме того, впервые был использован термин «timebox», он был создан Скоттом Шульцем и использован в DuPont. Команде предоставляется фиксированный временной интервал, в течение которого должна завершиться работа над продуктом. В рамках timebox «выполняется непрерывная итеративная разработка», в целях создания рабочей системы [13]. Мартин рекомендовал timebox продолжительностью 60 дней для команды из 1-5 человек. Методология временных рамок оказалась успешной, так как все проекты были завершены за меньшее время.

RAD нацелен на быструю поставку, итеративную разработку, небольшую команду высококвалифицированных разработчиков и участие пользователей на каждом этапе. Данные, что эти идеи лежат в основе Agile-методов. Однако термин «timebox» используется

в Agile по-другому. В RAD это целая фаза, состоящая из множества итераций, а в Agile timebox означает время на одну итерацию.

В таблице ниже перечислены принципы Agile и их источники.

Таблица 1. Современные принципы Agile и их первые упоминания в истории развития гибких методологий

Принцип Agile	Источник
Высшим приоритетом является удовлетворение потребностей клиентов за счет своевременной и непрерывной поставки программного обеспечения	Первый принцип EVO: сделать рабочий продукт для реального конечного пользователя [12]
Поддержка изменения требований даже на поздних стадиях разработки	Новый принцип. Проблема изменения требований не имела решения до Agile
Частая поставка ПО за сроки от пары недель до пары месяцев	Принцип содержится в методологии EVO и RAD
Клиенты и разработчики должны работать совместно на постоянной основе на протяжении всего проекта	Относительно новый принцип, хоть и некоторые подходы до Agile рекомендовали взаимодействие с клиентом, идея постоянного общения является новой.
При создании ПО необходимы замотивированные и талантливые специалисты, для которых важно обеспечить комфортную среду, дать поддержку и доверить выполнение работы.	Эти идеи впервые были опубликованы в 1985 году в книге по психологии компьютерного программирования. Автор подчеркивает важности мотивации, а также упоминает о комфортной среде для работы программистов [14].
Самый действенный метод передачи информации — это личное общение	Вышеупомянутая книга [14] освещает то, как рабочее пространство может влиять на социальное взаимодействие, которое, в свою очередь, влияет на эффективность в работе. Автор акцентирует внимание на том, как коммуникации помогают передавать и получать полезную информацию и делиться опытом.
Agile способствует устойчивому развитию. Разработчики, клиенты и прочие заинтересованные стороны должны иметь возможность поддерживать постоянный темп развития.	В книге «Марш смерти» Эдвард Юрдон акцентирует внимание на важности управления и контроле прогресса и предлагает концепцию «daily build» для достижения успеха в развитии. [15]
Проведение ретроспектив на регулярной основе позволяет команде понять, как стать более эффективными в работе и улучшить производительность	Идея постоянного улучшения процесса разработки была представлена в Agile Meets CMMI, где говорилось, что над процессами должны работать не только менеджеры, но и команда разработчиков. [16]

Основное различие между гибкими и классическими методами управления связано с отношением к неопределенности, возникающей в ходе реализации проекта. В то время как в классическом управлении проектами любые изменения нежелательны, риски пытаются спрогнозировать и нивелировать с помощью специальных инструментов, для проектов, осуществляемых с применением гибких методов управления, изменения – возможность модернизировать конечный продукт так, чтобы он максимально отвечал потребностям клиентов и текущей рыночной ситуации.

2. Обзор гибких методологий разработки

2.1. Фреймворки Agile

Сегодня под Agile в широком смысле понимается не только объединение ряда гибких методов и инструментов управления, но и культурные изменения в организации, которые отвечают единой философии, базирующейся на четырех ключевых ценностях Agile Манифеста. К таким культурным изменениям относится отказ от командно-административных методов управления, развитие культуры доверия и взаимопомощи, поощрение креативности и нестандартного мышления и т.д. В узком же смысле Agile представляет собой ряд практик и методов, характеризующиеся определенными отличиями.

Agile следует 4-м основным принципам:

- Коллектив и взаимодействие в нем важнее механизмов работы.
- Результат важнее документации.
- Взаимодействие с заказчиком важнее согласований условий контракта.
- Готовность к изменениям важнее следования первоначальному плану.

Гибкая методология управления ИТ- проектами представляет собой набор методов разработки программного обеспечения, которые способствуют адаптивному планированию, эволюционному проектированию и внедрению, постоянному совершенствованию и своевременному завершению работ.

Гибкие методологии разработки схожи между собой, однако каждая методология имеет свои характерные черты. В таблице ниже представлены основные гибкие методологии разработки и их характеристики.

Таблица 2. Гибкие методологии разработки

№	Методология	Год создания	Автор	Характеристики
1	Crystal Method	1992	Alistair Cockburn	Метод разработан для команд от 6 до 8 человек, работающих над разработкой некритичных программных продуктов. Метод включает 3 обязательные и 4 дополнительные практики. Обязательные практики: 1. Частые поставки. 2. Рефлексия. 3. Личные коммуникации. Дополнительные практики: 1. Чувство безопасности. 2. Фокусировка. 3. Доступность экспертов. 4. Качественное техническое обеспечение.
2	Refactoring	1993	Martin Fowler	Основная идея – упростить систему (в нашем случае проект) без изменения ее функциональности. Упрощенной системой легче управлять: контролировать, следить за изменениями. Также простота системы создает условия для взаимозаменяемости сотрудников, что также является важным принципом agile.

№	Методология	Год создания	Автор	Характеристики
3	Dynamic System Development Method	1994	DSDM Consortium	Включает в себя принципы двух методологий, описанных выше, и подходит для реализации проектов, критичных с точки зрения безопасности. Также ряд новых принципов появляется в данном методе: обратимость изменений, наличие высокоуровневых требования на момент начала проекта, и рассмотрение тестирования как интегрированной части ЖЦ разработки, впервые упоминается о необходимости высокой степени профессионализма участников команды проекта.
4	Function-Driven Development	1997	Jeff De Luca	Данный метод был разработан для Сингапурского банка. Его ключевое отличие состоит в том, что он позволяет организовывать команды от 15 до 50 человек. Это обеспечивается благодаря разделению продукта по функциям на несколько частей и организации команд для разработки этих частей.
5	eXtreme Programming	1999	Kent Beck, Martin Fowler, Ward Cunningham	Состоит из двух видов практик: 1. Управленческие (частые небольшие релизы, вовлечение заказчика) 2. Инженерные практики (рефакторинг, разработка через тестирование и непрерывная интеграция)
6	Continuous Integration	1999	Kent Beck, Ron Jeffries (Grady Booch)	Суть данного метода состоит в непрерывной интеграции. Необходимость в ней вызвана тем, что на стадии соединения частей разрабатываемой системы могут быть выявлены ошибки и противоречия, работу придется переделывать, а это может занять большое количество времени.
7	Agile modeling	2001	Scott Ambler	Данный метод используется как дополнительный на сегодняшний день (к примеру, как часть DSDM или eXtreme Planning). Представляет собой набор принципов, объясняющих как команда должна взаимодействовать между собой и с заинтересованными сторонами, а также о необходимости документального оформления взаимоотношений с заинтересованными сторонами, что очень важно и плохо проработано в других методологиях.
8	Test-Driven Development	2002	Kent Beck	Основное отличие данного метода от других состоит в том, что после определения цели определяются пути измерения ее достижения (критерии приемки или тест, как это называется в данном методе), только поле того, как тест определен, запускается процесс дальнейшего планирования. В план работ попадают только, те работы, которые позволят системе, результату проекта пройти тест.
9	Lean Software Development	2003	Mary Poppendieck, Tom Poppendieck	Принципы: • Исключение потерь. • Акцент на обучении. • Принятие решений не на основе предположений и прогнозов, а после открытия существенных фактов. • Предельно быстрая доставка заказчику. Короткие итерации.

2.2. Феймворк Scrum

Фреймворк в рамках концепции Agile характеризуется как процесс разработки программного обеспечения в контексте реализации концепции Agile-менеджмента, предназначенный добавить энергии, внимания, ясности и прозрачности проектным командам, разрабатывающим программные системы.

Scrum – это самый простой и распространенный подход, который сводится к набору правил самоорганизации команды. В scrum- команде должен быть product owner, команда разработчиков и scrum-мастер. Scrum основан на построении мультидисциплинарных, самоорганизующихся небольших команд, которые планируют работать в краткосрочной перспективе – спринте, в результате которого продукт будет готов для демонстрации клиенту или другим заинтересованным сторонам.

Сам фреймворк Scrum подразумевает под собой разделение всего процесса создания ИТ-продукта на равные по длительности части, спринты, в течение которых должны быть достигнуты определенные результаты в отношении проекта, которые были поставлены перед спринтом.

Перед началом спринта организуется командное совещание и составляется список задач для Backlog спринта. На этой встрече оценивается время, необходимое для выполнения каждой задачи, чтобы понять, сколько задач будет уложено в один спринт, и обсуждается взаимодействие между всеми членами команды. Количество задач в бэклоге не ограничено, но время, необходимое для их выполнения, должно строго соответствовать времени, отведенному на спринт. В конце спринта организуется встреча для обсуждения результатов. Спринты очень удобно сравнивать друг с другом, что дает возможность управлять эффективностью работы [17].

Еще одной отличительной чертой фреймворка Scrum являются daily meetings – короткие каждодневные встречи, на которых обсуждается прогресс за прошедший день и ставятся цели и задачи на следующий день.

Для управления реализацией ИТ-проектов данный подход помогает четко структурировать все работы и задачи на ближайший период времени, а также сделать открытым процесс коммуникации между аналитиками, таксировщиками, разработчиками и заказчиком. Такой подход позволяет выдавать каждому спринту свой определённый функционал, который ложится в итоговую версию ИТ-продукта и который с каждым разом добавляет все больше ценности ИТ-решению.

2.3. Масштабирование гибких практик управления

Помимо понимания работы по Agile внутри команды необходимо добиться эффективного взаимодействия между ними. Для этого существуют различные способы (фреймворки) масштабирования гибких методов управления в организации.

Первый и наиболее распространенный способ – применение одного из традиционных фреймворков масштабирования: масштабированный гибкий фреймворк (Scaled Agile Framework), масштабированный скрам (Large-Scale Scrum), скрам над скрамом (Scrum of Scrums) и другие.

Наиболее популярным фреймворком для небольшой организации является «скрам над скрамом» (SoS). По степени популярности уступает лишь SAFe: применялся в 27 организациях из 100.

Следующим фреймворком является масштабированный скрам (LeSS), и он также основывается на масштабировании скрама, но в отличие от «скрама над скрамом» он предназначен для более крупных организаций, поскольку помимо ежедневных межкомандных совещаний включает в себя дополнительные встречи представителей команд в преддверии и по окончанию очередного спринта. На этих встречах представители договариваются об общем направлении очередного спринта, обсуждают возможности взаимодействия и координируют даты релизов. Несколько команд, разрабатывающих схожий функционал, объединены в продуктовые группы с единым владельцем продукта. Несколько продуктовых групп, в свою очередь, включены в одно продуктовое направление. Таким образом, в данном фреймворке управленческие слои сведены к минимуму, многие важные решения команды принимают самостоятельно.

Масштабированный гибкий фреймворк (SAFe) является наиболее популярным способом масштабирования гибких методов управления: применялся в 28% случаев. Он применим для наиболее крупных организаций, поскольку использует более структурированный подход к трансформации. Каждая команда работает по скраму в течение двухнедельного цикла. Для координации деятельности команд, работающих над связанными задачами, создан уровень Программы. На этом уровне все задачи подразделяются на 2 группы: функциональные эпики и архитектурные эпики. В течение пяти спринтов команды совместно работают над выпуском конечного продукта или реализацией конкретного проекта. Для управления проектами в рамках одного направления создан уровень Портфеля, позволяющих топ-менеджерам компании выделять приоритеты разработки и распределять организационный бюджет. Масштабированный гибкий фреймворк отличается высокой структурированностью и предназначен для организаций, где традиционный метод укрепился наиболее сильно, поскольку

в нем достаточно много командного управления и планирования по сравнению с другими фреймворками.

3. Анализ подходов и практик, основанных на ценностях гибкой разработки программного обеспечения

3.1. Эффективность применения гибких методологий

Согласно экспертам Bain & Company, наибольшая эффективность применения гибких методологий достигается в ситуациях, когда рынок нестабилен, клиенты заинтересованы в постоянном сотрудничестве и не имеют изначально ясного видения конечного продукта, сам продукт является достаточно инновационным, при этом работу над ним можно разбить на отдельные этапы, а ошибки команды могут навести на ценные выводы.

При этом, нельзя не отметить тот факт, что на выбор между классическим и традиционными методами управления оказывают серьезное влияние особенности отрасли и внутренние возможности компании, такие как доступный технологический процесс, квалификация специалистов, наличие экспертизы в компании и возможности получения прибыли.

Согласно компании The Standish Group, представители которой в отчете The CHAOS Manifesto (The Standish Group, 2017) опубликовали результаты своего исследования, в котором видно, что гибкое управление примерно в три раза эффективнее традиционной каскадной модели управления IT-проектами.

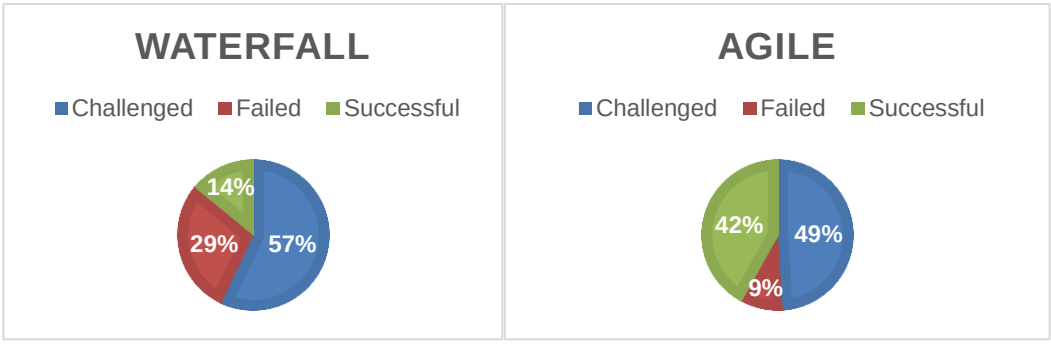

Рис. 1. Сравнение результатов применения традиционных и гибких методологий управления

Важно отметить, что успех гибких методологий, о котором говорится в отчете, является относительным, так как под “Successful” понимаются проекты были завершены в рамках бюджета и стоимости или с минимальными допустимыми отклонениями, под “Challenged” – проекты, которые были просто завершены, но с отклонениями от запланированных расписания и бюджета и под “Failed” – проекты, которые были закрыты без достижения поставленных целей и получения желаемых результатов. Таким образом, становится понятно, что только 42% проектов, использующих гибкие методологии по-

настоящему успешные, тогда как оставшиеся 58% проектов могут иметь сомнительный результат. Причин для этого может быть множество: плохое понимание принципов agile, низкий уровень зрелости команды и т.д.

Agile набирает популярность и с каждым годом охватывает большее количество отраслей. Рынок имеет свойство постоянно меняться и растет необходимость в Agile специалистах, внедряющих гибкие методологии в проекты. Компании становятся все более адаптивными к постоянно изменяющимся условиям, что является заслугой гибких методов.

Согласно исследованию Scrum Trek в России в 2020 году Agile все больше проникает в новые отрасли, как видно из рис. 1, среди них 42% в информационные технологии, 18% – в финансы, 8% – в тяжелую промышленность, 7% – в торговлю и другие. [18]

Согласно исследованию эффективности внедрения, Agile заметно ускоряет поставку продуктов – на 47% на этапе пилотирования Agile, на 60% – на этапе трансформации, на 75% – после полного внедрения Agile.

Обычно от Agile бизнес ожидает скорость, точнее, ускорение поставки (delivery) и выхода продуктов на рынок. Этой цели достигают больше половины участников исследования – 61% в 2020 году. Два наиболее заметных улучшения, которые каждый год отмечают люди после перехода на Agile – это управление часто меняющимися приоритетами и прозрачность работ.

2 эффекта от Agile в реальности не превосходят ожиданий во всех отраслях: снижение затрат и повышение качества продуктов. Такая статистика наблюдается уже несколько лет, что говорит об отсутствии влияния Agile на качество продукта, а также о больших затратах на внедрение и использовании Agile в России, т.к. проект, управляемый гибкой методологией требует большего количество средств, чем строго запланированный.

3.2. Статистика использования

Согласно исследованию Scrum Trek общемировые данные по целям и достижениям Agile — примерно такие же, как и в России:

- По достижению цели ускорения поставки в этом году Россия выровнялась с мировыми показателями: 60% в мире, 61% в России. Причем в России это – превышение ожиданий: цель повышения скорости в мире ставят 71% респондентов, в России – 55%.
- В списке реально достигнутых выгод первые 2 места в мире занимают те же показатели, что и у нас: 70% в мире, 78% в России и (65% в мире, 75% в России).
- Качество тоже является значимой целью (пятой по популярности в мире), но эта цель – как и в России – реализуется редко относительно ожиданий.

В мире Scrum используют более 58 % ИТ-организаций. Ежегодное исследование технологий Agile в российских организациях показало, что в стране подход управления проектами Scrum на постоянной основе используют более 48 % ИТ-организаций и более 43 % организаций экспериментируют с внедрением подхода в свою деятельность. Данные показывают, что российские ИТ-компании не переходят от Scrum к более сложным подходам даже спустя несколько лет после внедрения гибких методов. По мере развития опыта Agile процент компаний, использующих Scrum, растет с каждым годом. [19].

3.3. Подход к максимизации ценности Agile

Наблюдение и анализ только самого использования методологии не может дать верное представление о ее эффективности. Например, отслеживание скорости работы команды разработки ничего не говорит о том, действительно ли за заданное время команда предоставляет клиенту ожидаемый результат. Без измерения ценности эффективность любой гибкой методологии разработки основана только на предположениях, догадках и очевидных результатах деятельности.

Существует подход Evidence-Based Management (EBM), который представляет собой подход к управлению, основанному на фактических данных, которые дают возможность измерить, проанализировать и улучшить поставляемую ценность для клиента. Данный подход содержит в себе способы измерения и максимизации ценностей. Для управления ценностью необходимо разбить ее на метрики. Существует 4 показателя гибкости процесса:

1. Current Value — текущая ценность (CV).

   Цель CV - максимизировать ценность, предоставляемую клиентам в настоящее время; Показатель учитывает только текущую ценность, а не ту, которая может существовать в будущем. Метрики CV: Revenue per Employee, Product Cost Ratio, Customer Satisfaction, Usage Index.
2. Time-to-Market — время реализации (T2M).

   Выражает способность организации быстро предоставлять новые услуги или продукты. Ключевые метрики T2M: Build and integration frequency, Release Frequency, Mean Time to Repair, Cycle Time, Lead Time, Time-to-Learn
3. Ability to Innovate — готовность к инновациям (A2I)

   Выражает способность организации предоставлять новые продукты, которые могут в наибольшей степени соответствовать потребностям клиентов. Ключевые метрики

A2I: Usage Index, Innovation Rate, Defect trends, On-Product Index, Installed Version Index, Production Incident Trends, Time spent context switching.

4. Unrealized Value — нереализованная ценность (UV)

   Отображает потенциальную ценность, которая может быть получена, если организация сможет полностью удовлетворить потребности потенциальных клиентов. Ключевые метрики UV: Market Share, Customer or user satisfaction gap.

Заключение

Несмотря на то, что принципы Agile в целом новы, его идеи существуют давно, и люди, критикующие традиционные методы, предлагали альтернативные подходы, которые долгие годы не находили одобрения в мировом сообществе разработчиков. В первой главе были изучены исторические факты, подтверждающие, что недовольство негибкими подходами существовало задолго до 1990-х годов, некаскадные проекты были успешными еще в 1957 году, а также в 1980е годы были разработаны и успешно применены альтернативы негибким методам, такие как EVO, RAD и RIPP.

Таким образом, проанализировав гибкий подход к управлению проектами, можно сделать вывод, что в современных реалиях, когда бизнес постоянно должен подстраиваться под нужды клиента, требования к конечному ИТ-продукту постоянно меняются, для максимально успешного завершения ИТ-проекта больше пользы принесет именно применение методик гибкого управления проектом, так как этот подход позволяет, максимально оперативно и затрачивая наименьшее количество ресурсов, изменять ход своей деятельности и подстраиваться под изменяющиеся потребности клиентов. Гибкое управление проектами, управление на основе методологии Agile, наиболее подходит для ИТ-проектов, у которых очень размытые конечные желаемые результаты и нет возможности заранее определить все допущения и ограничения.

Основное отличие гибких методов управления от традиционных подходов состоит в циклическом (итеративном) процессе разработки и постепенном (инкрементальном) дополнении первоначального товара новым функционалом. Такой подход позволяет модифицировать товар в непосредственно в ходе разработки по мере появления новых требований или новой информации.

Рассмотрев фреймворки Agile, можно сказать, что каждый из них подразумевает под собой свои определенные условия для применения, определенные результаты и стиль работы команды, но все же наиболее применяемым и популярным фреймворком остается Scrum. Он

нацелен на контроль команды и определенные ограничения при реализации того или иного ИТ-проекта. Данный фреймворк хорош в своем варианте использования и приносит достаточно плюсов бизнесу и проекту, для того чтобы хотя бы задуматься о его применении в рамках своего проекта, ведь Scrum предоставляет значительно больше положительных моментов именно для ИТ-проектов и значительно упрощает систему контроля управления ИТ-проектами, помогает ИТ-командам в самоорганизации и выполнении коллективных обязательств.

Рассмотренные в работе примеры подтверждают, что методология Agile и входящий в нее фреймворк Scrum, позволяют более продуктивно достичь результатов и, следовательно, более качественно управлять ИТ-проектами, создавать качественные программные продукты не только в рамках ИТ-проектов, но и в других сферах деятельности, таких как продажи, банки, телеком и т.д.

Можно сказать, что правильное использование методологии Agile предоставляет при реализации ИТ-проектов следующие возможности:

- повысить гибкость команды;
- сократить цикл разработки продукта;
- вовлечь персонал в работу;
- делегировать полномочия по проекту членам команды;
- увеличить удовлетворенность клиента.

Масштабирование гибких методов управления в рамках подразделения или организации требует решения проблем координации и кооперации между командами. Для решения этих проблем существуют специальные фреймворки масштабирования. В то же время самостоятельно разработанные процессы и инструменты порой не уступают им в эффективности.

Также было определено, что лишь использование Agile и поверхностный анализ результат работы по гибкой методологии не может дать информации об эффективности использования. Для управления эффективностью команды необходимо использовать показатели, называемые метриками. Подход EBM дает возможность измерить, проанализировать и улучшить поставляемую ценность для потребителей и иных заинтересованных лиц. Данный подход содержит в себе способы измерения и максимизации ценностей.

Литература

1. Fowler M. The New Methodology, www.martinfowler.com.
2. Booch G. Object Solutions: Managing the Object-Oriented Project, Addison Wesley Longman Publishing Co., Inc., 1995

3. Royce, Winston, Managing the Development of Large Software Systems, IEEE Computer Society 25(6), 1970
4. Coad P., deLuca J. and Lefebvre E. Java Modeling Color with UML: Enterprise Components and Process with Cdrom, Prentice Hall PTR., IEEE Computer 21(5): 61-72., 1999
5. Rational Unified Process (RUP) [Booch G. (1995). Object Solutions: Managing the Object-Oriented Project, Addison Wesley Longman Publishing Co., Inc
6. Cockburn A. and Highsmith J. "Agile Software Development: The Business of Innovation." Computer 34(9): 2001, 120-127.
7. Larman C. Agile and Iterative Development: A Manager's Guide. C. Alistair and H. Jim, Pearson Education, Inc. 2004
8. Larman C. and Basili V. R. "Iterative and Incremental Development: A Brief History." IEEE Computer Society 36(6): 2003, 47-56
9. Royce, W.W., Managing the Development of Large Software Systems, Proceedings, IEEE WESCON, 1970, p. 1-9.
10. Gladden G. R. (1982). "Stop the Life-cycle, I Want to Get off." 7(2): 35-39.
11. McCracken D.D. and Jackson M.A. (1982). Lifecycle Concept Considered Harmful, ACM Press. 7: 29-32.
12. Gilb T. «Evolutionary Delivery versus the «waterfall model»» ACM SIGSOFT Software Engineering Notes 10(3): 1985, 49-61.
13. Martin J. Rapid Application Development, Macmillan Publishing Co., Inc., 1991
14. Weinberg, G.M., The Psychology of Computer Programming. 1985: John Wiley; Sons, Inc. 304
15. Yourdon, E., Death March: The Complete Software Developer's Guide to Surviving Mission Impossible Projects, ed. D.B. Paul. 1997: Prentice Hall PTR. 227
16. Turner, R. and A. Jain, Agile Meets CMMI: Culture Clash or Common Cause?, XP/Agile Universe 2002, LNCS 2418, 2002, p. 153-165.
17. Топ-7 методов управления проектами: Agile, Scrum, Kanban, PRINCE2 и другие. – URL: https://clck. ru/T6zJu
18. Сергей Рогачев. Отчет об исследовании Agile в России 2020. – [Электронный ресурс] // Научный блог. Электронный отчет. – Режим доступа: https://scrumtrek.ru/userfiles/reports/AgileSurvey20.pdf
19. Потапова, Е.Г. Навигатор цифровой трансформации: Agile-подход в государственном управлении: электронное издание / под ред. Е.Г. Потаповой. – М.: РАНхиГС, 2019. – С. 162.