БЕККЕР В. Ф., ПРИДЧИН К. А.
ПОВЫШЕНИЕ ЭФФЕКТИВНОСТИ УПРАВЛЕНИЯ ИТ-ПРОЕКТАМИ ЗА СЧЕТ AGILE И SCRUM

УДК 004.9+331.1, ВАК 05.13.10/2.3.4, ГРНТИ 50.01.75

Повышение эффективности управления ИТ-проектами за счет Agile и Scrum

В. Ф. Беккер, К. А. Придчин

Пермский национальный исследовательский политехнический университет, Березниковский филиал; г. Березники

Цель исследования – рассмотреть методологию Agile и фрейморк Scrum, а также их применение для повышения эффективности на примере компании «Крон», провести сравнение характеристик процессов ведения проектов в компании до и после применения. В данной статье рассматриваются основы методологии Agile и в частности фреймворк Scrum, а также проводится рассмотрение внедрения данных практик.

Новизна работы заключается в сложности и актуальности перехода компании от водопадной модели к гибким методологиям, в виду изменчивости бизнеса и возможности к быстрой адаптации к новым условиям.

Ключевые слова: *гибкие методологии, управление проектами, Agile, Scrum, Спринт.*

Improving the efficiency of IT project management by Agile and Scrum

V. F. Bekker, K. A. Pridchin

Perm National Research Polytechnic University, Berezniki branch; Berezniki

The purpose of the study is to consider the Agile methodology and the Scrum framework, as well as their application to improve efficiency using the example of the Kron company, to compare the characteristics of project management processes in the company before and after application. This article discusses the basics of the Agile methodology and in particular the Scrum framework, and also considers the implementation of these concepts in practice.

The novelty of the work lies in the complexity and relevance of the company's transition from the waterfall model to flexible methodologies, in view of the volatility of the business and the ability to quickly adapt to new conditions.

Keywords: *agile methodologies, project management, Agile, Scrum, Sprint*

Введение

Многие компании в настоящее время все активнее внедряют в свою работу информационные технологии, автоматизируют процессы. Из этого вытекает необходимость в найме хороших *IT*-специалистов, создание команд разработки.

В отличие от классических сфер – маркетинг, торговля, промышленность и множества других, управление проектами в *IT* довольно нетривиальный вопрос, ведь задачами в данном случае являются технические задачи, которые ведутся в цифровом виде. Также другим фактором сложности в управлении проектами в *IT* является то, что приходится разрабатывать и применять совершенно новые технологии, предметная область которых еще не столь изучена.

Управление проектами в *IT* представляет собой разработку новых продуктов, сопровождение старых продуктов, их доработку. При разработке новых продуктов, без опыта разработки в их предметной области, происходит то, что многие компании полагаются на старые, традиционные методологии управления проектами. Например, в водопадной модели, в которой разработка осуществляется циклами, сверху-вниз, сначала происходит проектирование, которое может затянуться на очень длительный срок, затем разработка и тестирование. Но так как в наше время наблюдается динамичность развития технологий, их применения и развития всех индустрий в целом, пока какая-либо компания разрабатывает свой проект в течение длительного времени, то что они делают может стать неактуальным, либо конкуренты выведут свой аналог гораздо раньше. Недостатки традиционных моделей управления проектами довольно легко устраняются применением гибких методологий, которые подстраиваются под текущие реалии изменчивости нашего мира.

Одной из самых популярных и эффективных методологий управления проектами является *Agile*. *Agile* представляет собой методологию управления проектами в сложной, запутанной среде, где требования бизнеса могут быть очень изменчивы.

Важным преимуществом *Agile* является более тесное сотрудничество, и совместная работа технических подразделений и бизнес-заказчиков. За счет этого решения принимаются очень быстро, официальных собраний гораздо меньше.

Исходя из вышеперечисленного, актуальность заключается в том, чтобы при управлении проектами в *IT*, разработка велась в наиболее тесном сотрудничестве с бизнес заказчиком с использованием гибких методологий разработки, что позволит выводить продукт в максимально короткий срок, снижая финансовые потери и повышая конкурентоспособность.

Понятие *Agile*

В последнее время многие говорят об *Agile*, называя этот подход инновационным. Команды, использующие *Agile*, быстрее достигают результатов, нежели те, кто использует классические процессы. Клиенты более удовлетворены результатами работы гибких команд, а также сами члены этих команд получают большее удовлетворение от своей работы. Применение гибких методов изменило область разработки программного обеспечения.

Agile – набор практик и методов для управления проектами в сложной (запутанной) среде. В русскоязычных книгах и статьях *Agile* называют гибкой методологией разработки, он создан как обобщение разных подходов разработки к ПО [0].

Agile используется для:

- ускорения вывода продукта на рынок;
- управления изменениями в приоритетах;
- улучшения взаимодействия между бизнесом и *IT*. Каждая сторона может говорить на своем языке и довольно часто при взаимодействии могут возникать проблемы.

Если говорить о разновидности фреймворков, то самая часто используемая вариация *Agile* — это *Scrum*.

Scrum

Рассмотрим более подробно методологию *scrum*.

Scrum – метод, базирующийся на *Agile*, в котором работа над проектом разбивается на спринты (итерации) – короткие, одинаковые по времени итерации. Численность команды, как правило, не более 10-12 человек. Команда состоит из *PO* (*product owner*), команды разработки и *SM* (*scram master*). *PO* – куратор группы, представитель бизнеса. Он следит за тем, чтобы конечный продукт отвечал его целям и задачам. *Scrum* мастер - это человек, который запускает *scram* в команде, следит за выполнением его правил, производит паркинг появляющихся проблем [2].

Scrum как фреймворк базируется на том, что самоорганизующиеся команды поставляют продукты в фиксированные сроки, которые называются «спринтами» [3, 4].

Основные характеристики *Scrum*:

1) самоорганизующиеся команды;
2) продукт разрабатывается серией «спринтов», которые длятся не более месяца;
3) требования записываются в виде единого списка – «бэклога продукта»;
4) инженерные практики не являются частью *Scrum*;
5) использует простые правила для создания гибкой среды разработки проектов.

Фреймворк *Scrum* состоит из *Scrum* команд и связанных с ними ролей, мероприятий, артефактов и правил [5]. Каждый элемент фреймворка служит определенной цели, и является ключевым для успеха и использования *Scrum*.

Scrum основывается на теории управления эмпирическими процессами. Эмпиризм утверждает, что знание приходит с опытом, решения принимаются на основании того, что является известным [6].

Scrum предписывает четыре формальные возможности для инспекции и адаптации:

- планирование спринта;
- ежедневный *Scrum*;
- обзор спринта;
- ретроспектива спринта.

Если говорить о ролях, то *Scrum* команда состоит [8]:

1) владелец продукта (Product Owner) – апологет продукта, который понимает его ценность для бизнеса. Также он доносит потребности кастомера/стекхолдера до команды разработки, но не отвечает за техническую сторону процесса. Также он отвечает за пользовательские истории и определяет их приоритет;

2) *Scrum*-мастер – отвечает за соблюдение правил фреймворка, за проведение церемоний. Помогает владельцу продукта и команде разработки устранять препятствия и отвлекающие от работы факторы. Вся основная коммуникация людей вне команды проходит при участии непосредственно *Scrum*-мастера;

3) команда разработки отвечает за выполнение всех технических задач, связанных с разработкой. Команда как правило, кроссфункциональна и в зону ответственности входит дизайн, программирование, тестирование и так далее. При этом команда выполняет все вышеперечисленное, руководствуясь пользовательскими историями и их приоритетностью.

В ходе *Scrum* процесса формируются следующие артефакты [9]:

1) бэклог продукта – это упорядоченный список всего, что может быть нужным в продукте, он является единственным источником требований для любых изменений, которые может потребоваться внести в продукт;

2) инкремент – это сумма всех выполненных требований бэклога продукта, реализованных во время текущего спринта, и ценности всех предыдущих спринтов. По окончанию спринта новый инкремент должен быть «готовым», то есть он должен быть пригодным к использованию и отвечать определенным *Scrum* командой критериям «Готовности»;

3) элемент бэклога продукта – задачи, стори, которые необходимо выполнить за спринт. Как правило декомпозируется на несколько задач;

4) цель спринта – то, что необходимо сделать, чтобы выполнить элемент бэклога продукта. Целью спринта является измеримая, достижимая цель, которая не является технической задачей;

5) Burn-down чарт – работа, которая остается до полного выполнения задач спринта. Может быть, как восходящим, так и нисходящим, в зависимости от того, с чем команда сталкивается при выполнении задачи. Burn-down не является отчетом о продвижении команды, а методом определения трудностей для поддержания активности.

Основой *Scrum* является спринт длительностью в один месяц или менее, в течение которого создается потенциально готовый к выпуску и использованию инкремент продукта. Лучше, когда длительность спринтов является постоянной на протяжении всего периода разработки. Следующий спринт начинается сразу же по окончании предыдущего.

Спринты состоят [10] из планирования спринта, ежедневных *Scrum*, разработки, обзора спринта, а также ретроспективы спринта.

Во время спринта:

- нельзя вносить какие-либо изменения, которые бы были угрозой для цели спринта;
- цели по качеству продукта детерминированы;

- объем работ может быть уточнен и повторно обговорен между владельцем продукта и командой разработки по мере накопления знаний.

Планирование спринта (*Spring planning*) – в нем учувствует вся *Scrum* команда. В течение этого мероприятия *PO* предлагает задачи из бэклога, расставляет приоритеты, и команда берет их в работу.

Цель Спринта – это задача, которая будет достигнута в результате спринта благодаря реализации бэклога продукта, и которая объясняет команде разработки, почему она разрабатывает.

После того, как цель спринта определена, команда разработки решает, каким образом воплотить функциональность в «готовом» инкременте продукта. Элементы бэклога продукта, взятые в спринт вместе с планом для их разработки, называются бэклогом спринта.

Ежедневные *Scrum* (*Daily*) – это 15-минутные мероприятия для команды разработки с целью синхронизации действий и создания плана работы на ближайшие 24 часа. Это делается для инспекции проделанной работы с момента прошлого ежедневного *Scrum* и прогноза того, что может быть сделано до следующего. Эти совещания проводятся в одном и том же месте, в одно и то же время для уменьшения путаницы.

1) Обзор спринта (Демо) - встреча по обзору спринта проводится в конце спринта для инспекции инкремента и при необходимости адаптации бэклога продукта. Во время обзора спринта *Scrum* команда и заинтересованные лица обсуждают выполненную во время спринта работу.

2) Ретроспектива - ретроспектива спринта дает *Scrum* команде возможность инспектировать себя и создать план улучшений для следующего спринта. Ретроспектива происходит после обзора спринта, перед последующим планированием спринта. Это ограниченная тремя часами встреча для одномесячного спринта. Для более коротких спринтов обычно выделяется меньше времени.

Применение *Agile* методологии на примере организации ООО «Крон»

Данная *IT* компания-разработчик "Крон" с 2008 года занимается оказанием различных услуг, в сфере информационных технологий, для клиентов по всей России, разработкой и внедрением *IT*-решений для малого и среднего бизнеса. Компания имеет полный цикл: разработка, продажи, и поддержка программного обеспечения для конечного клиента [11].

Изначально в компании процесс разработки осуществлялся следующим образом: руководитель проекта, на основании собственных суждений, планов и идей составлял список функций, которые затем предлагались клиентам. После этого функционал подвергался прототипированию и показывался команде, как задача на данный момент времени. Важным нюансом являлось то, что команде фактически давался готовый результат, который они должны были принять. Предлагалось только лишь обсуждение деталей и частных функций, без комплексной оценки разрабатываемой версии.

Такой подход напоминает водопадную модель, только заказчиком является руководитель проекта. Также о соответствии водопадной модели говорит то, что

идет фиксация конечного набора функций, до этапа его разработки, после чего изменения и коррекция не производится. Собирается набор функций, утверждается и после этого на разработку тратиться значительное количество времени, затем только производится презентация клиентам и собирается обратная связь. Минусы этого подхода, это длинный цикл разработки, недостаточная проверка на целесообразность той или иной функции, получение обратной связи только на финальном этапе. Получение обратной связи только на финальном этапе приводит к издержкам, в частности та или иная фича может оказаться клиентом невостребованной и приводит к увеличению платежей.

Кроме обычных и известных проблем, такой подход выявил еще одно узкое место, это отдел разработки. Узким местом в данном случае является то, что отдел разработки очень загруженный участок бизнес-процесса, его участие в разработке неактуальной фичи (функции) приводит к том, что те или иные более важные фичи (функции) находятся в статусе паузы. Довольно часто случалось такая ситуация, когда клиенты получают ненужную и невостребованную им функцию, но критичные баги не исправляются, так как на них не было фокуса.

В результате основными причинами изменений в компании стало:

- увеличение количества разработчиков и снижении производительности;
- недостаток в управляемости процесса;
- срыв поставелнных сроков;
- ошибки при оценке строимости разработки;
- сложность поддержки;
- удлинение цикла исправления багов.

Проблемы, которые были в компании:

1. нет позиционирования продукта;
2. применение водопадной модели на меняющемся рынке – как следствие долгая разработка весьма сложного блока без проверки его клиентом;
3. недостаточность получения обратной связи от клиентов (работники считали, что знают, как надо лучше, чем клиенты);
4. наличие множества недоделанных функций (идей и фантазий достаточно, но ни одна не доводится до конца);
5. отсутствие внятной методики измерения производительности;
6. несоблюдение графика релизов, так как продукт сложный и разрабатывается длинными итерациями.

До начала внедрения гибких методик разработки, в рамках сервиса "Офис 21", вопрос рыночного позиционирования не поднимался и дополнительно не исследовался. Разработка и продажи велись, не исходя из потребностей рынка, а исходя из понимания что нужно клиенту руководителем компании. Данный подход уместен на низко конкурентных и растущих рынках, но, когда рынок стабилизируется, этот путь приведет компанию к гибели. Именно гибкие методики разработки ставят во главу угла пользователя и его задачи, а не компанию и ее видение. Благодаря проведенным интервью с клиентами было сформировано понимание какие именно проблемы решает сервис и что для клиента важно, а что нет. Полученная картина отличалась от того, что рисовала

команда и позволила сформировать продукт и позиционирование сервиса. Если в начале пути «Офис 21» - это про «Рост продаж и контроль сотрудников», то по итогу осмысления «Офис 21» - рост продаж существующим клиентам» [12].

• приоритизация разработки, с точки зрения уместности в позиционировании

До начала внедрения *Agile* очередность и приоритет разработки функционала осуществлял руководитель компании, на основании своего представления о конечном результате. Разработка часто была хаотична, и переходы между смежными модулями могли быть практически каждый день. После внесенных изменений очередь разработки выстраивается на основе коэффициентов, которые расставляются каждой карточке, а тематика разработки на неделю задается темой спринта, и задачи из другой темы, кроме исправления ошибок, в данном спринте не выполняются.

• внедрение клиентов в процесс определения функционала

До начала изменений мнение клиента практически не учитывалось – подход был такой, что команда знает лучше, что нужно рынку и как это продавать. После внедрения этих методик мнение клиентов стало приоритетным, и никакая функция не идёт в разработку до тех пор, пока она не получает подтверждения со стороны клиентов, или же предварительное согласие на её приобретение. Это позволило значительно улучшить продукт, с точки зрения внедрения у клиента, и получить дополнительный доход, снизить издержки.

• концепция *MVP*

Философия гибких методов разработки подразумевает то, что клиент должен быть допущен к тестированию и получению обратной связи в максимально сжатые сроки. В то время как в классической модели до клиентов доставляется конечная функция, которая протестирована, отлажена и полностью интегрирована в программное обеспечение. Был внедрен подход, при котором минимально работающая функция тут же допускается в работу, и клиенты начинают её тестировать и высказывать замечания. Это значительно экономит время на разработку, так как убраны из списка задач те, которые клиент не запрашивает, при первичном тестировании.

• минимизация и упрощение каждой функции, доработка после обращения клиента

До внедрения гибких методов разработки, любая функция после того как она была реализована, проходила тестирование, затем обкатку и доработку исключительно внутри компании. После внедрения agile подход изменился: функция, которая начала хоть как-то работать, тут же становится доступна клиентам и собирается обратная связь (что не работает, как должно быть, почему именно так). И только после этого она дорабатывается. Основная идея состоит в том, что клиенты гораздо лучше знают, как и что им нужно, но выразить конкретно они этого не могут - они могут показать на примере того, что не работает, или работает не так, как они хотят.

• внедрение покер-планирования на этапе оценки пожеланий и списка функционала

Это позволило учитывать мнение всей команды, а не только руководителя, при определении приоритетности разработки той или иной задачи на карточке. Это дает возможность гораздо быстрее реагировать, так как на передовой находится техническая поддержка, их мнение стало учитываться при выборе вектора продукта.

- продажа несуществующего функционала

Методика, заимствонная из *Customer Development*, позволила получать коммерческое подтверждение необходимости той или иной функции, до того, как начали его разрабатывать. В итоге это привело к тому, что разрабатываются только те функции, за которые клиенты готовы платить, или уже заплатили. Основная сложность внедрения данного подхода – объяснить менеджерам по продажам, что презентовать и продавать клиенту то, чего у тебя сейчас нет - это нормально, и компания сможет это сделать, только после того, когда клиент примет финансовое участие.

Изменения в разработке:

- миграция с *trello* на клабхаус

Первоначально сбор карточек с задачами и функциями вёлся на канбан доске в рамках сервиса *trello*, так как это наиболее удобный и простой сервис, ориентированный на широкий круг пользователей. Но в нём нет абсолютно никаких инструментов, ориентированных на Agile. В процессе внедрения встал вопрос о методике оценки и учёте трудозатратности карточки, и формирование системы отчётности, что привело к переходу на специализированные решения, которые аналогично построены вокруг канбан доски, но заточены под управление командой разработчиков.

- внедрение понятия спринта, сроком 1 неделя

Первоначально функция разрабатывалась ровно столько, сколько требовалось, выпуск релизов не был привязан ко времени или к какому-то интервалу, он был хаотичный. При внедрении *Agile*, была разработана и реализована концепция спринта, как рабочей недели, ориентированная на выпуск определённой функции, от момента её обдумывания до релиза. Это значительно повысило концентрацию разработчиков на определённой задаче, и привело к увеличению производительности, так как человек погружаюсь в процесс, дает больший результат, нежели, когда он переключается между множествами различных задач.

- внедрение карточки функции (*User Story*)

Раньше существовала карточка функции, в которой описывалось, что нужно реализовать, но там не фигурировало понятие «зачем и как это будет работать», что усложнило понимание пользовательской историй для конечного разработчика. Внедрение карточки из истории пользователя привели к тому, что программисты иногда дают интересные замечания, что и как можно реализовать, понимая контекст, который описан в карточке.

- внедрение классификации «ошибка», «функция», «мелочь»

Это инструментарий, который предлагает сама платформа, внедрение такого разделения всех задач позволило понимать, какое количество ошибок мы исправляем, или же мы работаем больше на наращивание функционала, или

делаем мелкие исправления. Это позволяет лучше планировать спринт, когда выделяется время исключительно на исправление ошибок.

- шкала оценки сложности функции (*estimate*)

Одно из важнейших приобретений от внедрения гибких методов разработки - инструмент, позволяющий провести количественную оценку сложности задачи + провести суммирование работы по итогам месяца. В рамках компании были утверждены следующие ориентиры, для установки значения *estimate:*

1. мелкий и понятный фикс - 1СП;
2. мелкий фикс, но надо немного подумать - 2СП;
3. добавление сущности, требуется миграция - 3СП;
4. серьезная переделка алгоритма, фронта, миграции - 5СП;
5. непонятная задача, требует серьезного времени на осмысление алгоритма, тестинг, кодинг - 8 СП.

- внедрение покер-планирования на этапе оценки сложности карточки

Подход, который снизил ошибки при оценке сложности, а значит и сроков разработки. Основное отличие в том, что ранее сложность оценивал каждый разработчик, исходя из своих знаний и опыта. После перехода на покер-планирование, оценки стали коллективными, что привело к тому, что некоторые моменты, ранее не учитываемые (со стороны технической поддержки, к примеру) стали влиять на итоговую оценку сложности задачи. В результате планирование стало точнее, и допуск оценки стал +-15% от намеченного срока.

- Ежедневные совещания *daily*

Внедрение регулярных (ежедневных) коротких совещаний привело к тому, что проблемы, которые кажутся незначительными, стали решаться быстрее. Особенно заметным этот момент является при работе с программистами – так как каждому нужно рассказать, что он делает и с какими сложностями столкнулся, то он вынужден это анализировать и проговаривать, в результате вынесенные на общее обсуждение проблемы очень часто решаются гораздо быстрее не методами разработки, а административными методами.

- Перевод оплаты разработчиков на шкалу *estimate*. Бонусы в случае выполнения и перевыполнения планов

Это уже, скорее, не относится к гибким методам разработки напрямую, но повлияли именно они на изменение трудовых договоров – ранее программисты работали по определённому графику, и получали оплату не за результат, а за отработанные часы, что крайне сложно контролировать, измерять, и в целом не говорит о эффективности сотрудника. После внедрения системы измерения сложности по карточкам логичным шагом стал перевод системы оплаты на данные рельса. Выглядит это достаточно просто – сумма, сделанная за месяц, умножается на стоимость единицы, и получается заработная плата, которую заработал данный человек за период. Это решение убрало проблему контроля, так как программисты сейчас не обязаны лишь отработать время, в их интересах добиваться результата, кроме того это упростило программистам финансовый рост, им сейчас не нужно обосновывать почему надо поднимать заработную плату, или наоборот, чем они занимались весь месяц.

- бот-морковка для борьбы со срывом сроков

После первоначального внедрения методики оплаты по количеству сделанного, компания столкнулась с маленькой проблемой – программисты не всегда понимают сколько не сделали за данный период, и часто срывали свои объемы, что приводило к снижению заработной платы за месяц. Либо они пытались переработать в последние несколько дней, что сказывается на качестве и результате. Было разработано следующее решение: составлен предварительный план на каждого сотрудника, и написана небольшая программа, которая каждый день в 12:00 уведомляет каждого сотрудника о том на каком сейчас он вместе с точки зрения выполнение плана. Это выглядит в виде сообщения в телеграмм, в котором говорится, что сделано за день столько-то, за месяц столько-то. И последнее предложение – вывод, всё ли идёт по плану, или он отстает/опережает на столько-то. После внедрения данного алгоритма программисты почти всегда стали выполнять отличный план по количеству сделанного.

Таким образом, разобрав подробно все изменения по части менеджмента и по осуществлению процесса разработки, можно провести параллель между тем, как в компании «было», и как теперь «стало» после применения *agile* методологии. Данные представлены в таблице 2.

Таблица 2. Результаты перехода на Agile

Процесс	**Было**	**Стало**
Оценка работы программиста	Затраченное время, со слов программиста	Сумма *estimate* сделанных карточек задач
Составление плана разработки	Решение руководителя	Покер-планирование с командой
Выбор функции для разработки	Решение руководителя	Коэффициент важности
Оценка стоимости доработки	Решение руководителя	Сумма *etimate* * на стоимость 1шт
Начисление заработной платы разработчикам	Решение руководителя	Сумма *etimate* * на стоимость 1шт
Обратная связь от клиента	Обращения в тех.поддержку	Периодические встречи, интервью
Исправление ошибок	Периодически, в выделенное время	В течении 1 дня, с момента обнаружения
Разработка крупных модулей	Сначала разработка - потом презентация	Сначала презентация - потом разработка
Цикл разработки	1-2 месяца	1 неделя

Заключение

В данном исследовании была рассмотрена методология *Agile*, а также основные принципы фреймворка *Scrum* и его применение на примере компании ООО «КРОН». В результате применения *Agile* и *Scrum* получилось достичь следующих улучшений:

1. разработка модуля начинается только после получения согласия на покупку от нескольких клиентов, что позволило не тратить ресурсы впустую и увеличить маржинальность;
2. оценка работы перешла из качественной в количественную форму, на основании *estimate* метрики;
3. выстроена обратная связь с клиентом, что позволяет разрабатывать более востребованный продукт;
4. разработчики понимают, как могут влиять на свой доход и из чего он формируется;
5. за счет уменьшения цикла разработки клиенты стали получать исправление ошибок и новый функционал быстрее, пусть и с некоторыми упрощениями.

Изначально поставленная цель была достигнута, удалось рассмотреть базовую теорию по Agile и Scrum, а также рассмотреть планомерный переход от водопадной модели к Scrum, произвести сравнение всех основных показателей процессов до и после.

Таким образом, применение технологии *Agile* приводит к совершенствованию деятельности компании, получению дополнительных возможностей и устранению части непроизводительных затрат. Поэтому на российском рынке наблюдается ежегодный прирост компаний, использующий рассматриваемый подход.

Но следует обратить внимание, что эффективная практика применения гибких методов основана на использовании техник креативного мышления, которая в последнее десятилетие является не просто востребованной, а реально формирующей вовлеченность персонала в бизнес–процессы компании. В условиях глобализации каждая компания стремится получить конкурентное преимущество, и основой для этого становится, прежде всего, персонал, вне зависимости от места в иерархии власти. Кроме того, происходит смена поколений, а, значит, прошлые технологии работы с персоналом уже неэффективны. Этим и объясняется востребованность гибкого подхода.

Список использованных источников и литературы:

1. Кон М. Скрам: Гибкая разработка ПО. – М.: ООО «И.Д. Вильямс», 2011. – 575 с.

2. Лоффлер М. Ретроспектива в Agile, Пер. с англ. – М.: ООО «Манн, Иванов и Фербер», – 2020. – 185 с.

3. Пихлер Р. Управление продуктом в Scrum. М.: ООО «Манн, Иванов и Фербер», 2017. – 230 с.

4. Затонский А. В. Теоретический подход к управлению социально-техническими системами // Программные продукты и системы. – 2008. № 1. – С. 29-32.

5. Близнюк А. М. Методология *Scrum* как инструмент управления // Perspective Research and development. – Петрозаводск: 2021, – С. 35-39.

6. Инюшин В. И. Agile-подход в управлении проектами // Управление проектами. Севастополь, 2017. – С. 60-65.

7. Кирьянов Б. А. Применение гибкой методологии Agile в управлении проектами // Техника и технологии: теория и практика. – Пенза, 2020. – С. 38-42.

8. Мелихова А. Е. Методология Scrum: возникновение, философия и принципы использования // Управленческие науки в современном виде, – 2018. № 1. – С. 198-204.

9. Затонский А. В., Варламова С. А. Информационное обеспечение поддержки принятия решений на примере составления расписания занятий образовательной организации // Вестник Южно-Уральского государственного университета. Серия: Компьютерные технологии, управление, радиоэлектроника. – 2018. Т. 18. № 3. – С. 88-106.

10. Цадурьян Э. Л. *Scrum* как самый популярный метод управления в проектном менеджменте // Наука и просвещение: –Пенза, 2019. – С. 41-45.

11. Официальный сайт *IT*-компании «Крон» [Электронный ресурс]. – Режим доступа: http://itcron.ru (дата обращения: 19.04.2022).

12. Официальный сайт SaaS системы «Офис 21 века» [Электронный ресурс]. – Режим доступа: https://of21.net (дата обращения: 19.04.2022).

List of references:

1. Cohn M. Scrum: Agile software development. - M .: LLC "I.D. Williams", 2011. - 575 p.

2. Loeffler M. Retrospective in Agile, Per. from English. - M .: LLC "Mann, Ivanov and Ferber", - 2020. - 185 p.

3. Pichler R. Product management in Scrum. M.: OOO "Mann, Ivanov and Ferber", 2017. - 230 p.

4. Zatonsky A. V. Theoretical approach to the management of socio-technical systems // Program products and systems. - 2008. No. 1. - P. 29-32.

5. Bliznyuk A. M. Scrum methodology as a management tool // Perspective Research and development. - Petrozavodsk: 2021, - P. 35-39.

6. Inyushin V. I. Agile approach to project management // Project management. Sevastopol, 2017. - S. 60-65.

7. Kiryanov B. A. Application of flexible Agile methodology in project management // Engineering and technology: theory and practice. - Penza, 2020. - S. 38-42.

8. Melikhova A. E. Scrum methodology: emergence, philosophy and principles of use // Management sciences in the modern form, - 2018. No. 1. - P. 198-204.

9. Zatonsky A. V., Varlamova S. A. Information support for decision-making on the example of scheduling an educational organization. Bulletin of the South Ural State University. Series: Computer technologies, control, radio electronics. - 2018. V. 18. No. 3. - P. 88-106.

10. Tsaduryan E. L. Scrum as the most popular management method in project management // Science and education: - Penza, 2019. - P. 41-45.

11. Official site of the IT-company "Kron", http://itcron.ru (date of access: 04/19/2022).

12. Official website of the SaaS system "Office of the 21st century", https://of21.net (date of access: 04/19/2022).