УДК 005.8

ЦЕЛЕСООБРАЗНОСТЬ ПРИМЕНЕНИЯ ГИБКИХ МЕТОДОЛОГИЙ В УПРАВЛЕНИИ ПРОЕКТАМИ НА ПРИМЕРЕ СТУДИИ МОБИЛЬНОЙ РАЗРАБОТКИ «65 ГИГАБАЙТ»

Михайлова Е.А.[1]

[1]*ФГБОУ ВПО «Ижевский государственный технический университет имени М.Т. Калашникова», Ижевск, estanovskih@65gb.net*

На сегодняшний день повышение эффективности управления проектами по разработке программного обеспечения приобретает особую значимость ввиду фактического превышения бюджета и срока проектов сверх запланированных. В настоящей статье рассматривается использование гибкой методологии Scrum как средство повышения эффективности деятельности на примере студии мобильной разработки «65 Гигабайт». Раскрыты понятия прогнозируемой и гибкой методологий в управлении проектами по разработке программного обеспечения. Проведен анализ каскадной и итеративной моделей жизненного цикла программного продукта. Описана сущность и преимущества Scrum методологии. Обоснован выбор студии мобильной разработки «65 Гигабайт» в пользу Scrum методологии. Предложен ряд мероприятий по внедрению гибкой методологии Scrum на предприятии. Определены достигнутые при помощи Scrum методологии результаты. Сделан вывод о целесообразности применения гибкой методологии Scrum.

Ключевые слова: управление проектами по разработке ПО, разработка мобильных приложений, прогнозируемые методологии, гибкие методологии, scrum.

APPROPRIATENESS OF AGILE METHODOLOGIES IN PROJECT MANAGEMENT ON CASE OF MOBILE DEVELOPMENT STUDIO "65 GIGABYTE"

Mikhailova E.A.[1]

[1]*Izhevsk State Technical University of M.T. Kalashnikov, Izhevsk, estanovskih@65gb.net*

Nowadays improving the efficiency of project management in software development is particularly significant in view of the actual cost overruns and project over the planned period. This article reviewed the use of Agile Scrum as a means as a mean to improve the efficiency of the example of mobile development studio "65 GB". The analysis of the cascade and iterative models of software lifecycle was made. The nature and advantages of the Scrum methodology were described. The choice of mobile development studio "65 GB" in favor of the Scrum methodology was justified. A number of measures to introduce Agile Scrum in the enterprise was requested. Achieved using Scrum methodology results defined. Results achieved using Scrum methodology were defined. The conclusion about the appropriateness of Agile Scrum was made.

Keywords: project management, software development, mobile application development, predictable methodology, agile, scrum.

Начиная с самых первых программных проектов и по нынешний день, разработка программного обеспечения была и остаётся мало предсказуемым и далеко не всегда успешным делом. Значительный процент проектов по созданию программного обеспечения по-прежнему заканчивается с превышением бюджета, сроков, а созданные в результате программы часто не до конца отвечают требованиям пользователей либо приносят мало реальной пользы бизнесу. Перечисленные проблемы являются основными проявлениями, так называемого кризиса программного обеспечения. Несмотря на значительные интеллектуальные усилия, потраченные на поиск способов преодоления кризиса, до сих пор так и не найдено сколько-нибудь универсальное решение[2].

Вероятно, наиболее заметным изменением последних лет в области процессов разработки программного обеспечения стало появление слова agile. Все говорят о гибких

методах разработки, о том, как сделать гибкой команду разработчиков, или о том, как противостоять надвигающейся армии приверженцев agile, готовых растоптать устоявшиеся годами практики разработки[4].

Прежде чем приступить к предмету гибкой методологии, целесообразно обратиться к генезису управления проектами по разработке ПО (программного обеспечения).

Ранние подходы к разработке были мало формализованы и представляли собой процесс, который принято называть Code-and-Fix (кодирование и исправление). При таком подходе разработка программного обеспечения начинается непосредственно с кодирования (без предварительного планирования, анализа требований и проектирования). После этого найденные в коде проблемы (дефекты, несоответствие требованиям и т.п.) исправляются путём внесения множественных изменений в код. Неудивительно, что после некоторого количества таких изменений система становится запутанной, её сложно поддерживать и расширять. Со временем стало понятно, что для создания больших устойчивых программных систем нужны более продуманные и формальные подходы. В поисках решения проблемы внимание было обращено на более зрелые к тому времени области человеческой деятельности, связанные со сложным производством - прежде всего, на системотехнику (systems engineering), а также другие инженерные дисциплины, такие как проектирование и строительство мостов, сооружений и т.д. В результате попытки использовать проверенные в других областях инженерные методы для разработки программного обеспечения появилась новая дисциплина - программная инженерия (software engineering), а в качестве фактического стандарта на долгие годы утвердились так называемые инженерные методологии разработки программного обеспечения. Эти методологии также часто называют основанными на плане (plan-driven), потому что в их основе лежит предположение о том, что процесс разработки программного обеспечения является детерминированным инженерным процессом, который можно спланировать от начала и до конца и выполнить в соответствии с планом, используя формальные инженерные подходы[2]. В качестве основного варианта построения жизненного цикла программного продукта использовалась каскадная модель.

Каскадная модель определяет разработку ПО как строгую последовательность этапов, при этом переход на следующий этап происходит только после полного выполнения работ на текущем этапе.

При использовании каскадной модели основными этапами, составляющими жизненный цикл процесса разработки ПО, являются: анализ предметной области; проектирование программно-аппаратного комплекса; кодирование; тестирование; внедрение и сопровождение.

Анализ заключается в получении и формализации сведений об исходном объекте автоматизации. Часто, особенно для анализа сложных объектов, используются модели, получаемые путем применения методов системного анализа. На этом этапе определяются требования к будущей программной системе, объемы проектных работ, риски, необходимые трудозатраты, формируется техническое задание и план-график выполнения работ.

Проектирование заключается в создании модели будущей программной системы. Основными представлениями при этом являются: архитектура программной системы; модульная структура программной системы; алгоритмы функционирования; структуры данных; потоки входных и выходных данных программной системы.

Исходными данными для проектирования являются модели и документация, получаемые в результате анализа. Кодирование заключается в создании текстов программ на целевом языке программирования, в точности соответствующих моделям, полученным в результате проектирования. Результатом кодирования являются исполняемые модули программной системы.

Тестирование заключается в проверке работоспособности программной системы в различных режимах ее использования с целью выявления возможных дефектов в функциях, логике и форме реализации.

Внедрение и сопровождение заключается в передаче программной системы заказчику и возможном последующем изменении эксплуатируемых программ с целью исправления ошибок, адаптации к изменениям внешней среды, усовершенствования по требованиям заказчика.

Использование классического цикла разработки программного обеспечения позволяет упорядочить процесс разработки, получить план и временной график выполнения работ. В то же время такая модель имеет существенный недостаток: цикл основан на предположении о точной формулировке требований к программной системе. В реальности в начале проекта требования определяются лишь частично. Так как заказчик имеет доступ к результатам проекта только по окончании выполнения всех работ, он не может уточнить требования и в большинстве случаев получает совсем не тот продукт, который ему был необходим. Именно этот недостаток сделал классический цикл разработки программного обеспечения практическим неприменимым.

Как показывает практика, во многих случаях заказчик не может четко сформулировать требования к программному продукту, который он хочет получить. Более того, зачастую сам процесс разработки и внедрения программных систем влияет на внутренние процессы объекта автоматизации и может изменить их таким образом, что первоначальные требования к программному продукту потеряют смысл[3].

Переход к инженерным методологиям и каскадной модели, несомненно, был шагом вперёд. Он внёс определённую упорядоченность и организованность в процесс разработки, а также исключил многие проблемы подхода code-and-fix. Однако инженерные методологии по определению не могли решить всех проблем программных проектов, прежде всего, из-за заложенного в них внутреннего конфликта. Дело в том, что они изначально создавались по образцу других инженерных дисциплин и не до конца учитывали особенности, присущие именно программному обеспечению как уникальному виду продукции. Конфликт уникальности программного обеспечения с подходами инженерных методологий имеет, как минимум, два основных проявления:

Первое проявление заключается в недооценке роли человеческого фактора. Инженерные методологии рассматривали процесс разработки как последовательность шагов, выполняемую абстрактным исполнителем, т.е. основное внимание фокусировалось на том, что нужно сделать в процессе разработки, а не на том, кто это будет делать. В действительности же большинство проблем программных проектов имеют социальную, а отнюдь не технологическую природу. Именно участники проекта и отношения между ними являются основным фактором, влияющим на успех проекта.

Второе проявление состоит в несоответствии каскадного жизненного цикла природе программного обеспечения. Принципиальным моментом в данном случае является уникальность и новизна практически любого программного продукта. Разработка ведётся в условиях высокой неопределённости, и попытка учесть все факторы в начале проекта (составить детальные требования, затем полностью разработать дизайн системы и т.д.) заранее обречена на провал. Самое неприятное заключается в том, что недостатки водопадной модели не только затрудняют процесс разработки сами по себе, но вдобавок ещё и угнетают человеческие отношения. Например, создание и утверждение детальных требований в начале проекта часто приводит к противостояниям между заказчиком и исполнителем в случае необходимости внесения изменений на стадии реализации. Споры вокруг потенциальных изменений ведут к ухудшению отношений и разрушают слаженное взаимодействие команды с заказчиком. Но следует отметить, что существуют проекты, для которых водопадный жизненный цикл может неплохо работать, однако для большинства проектов упомянутые выше проблемы являются актуальными и в большом количестве случаев приводят к провалу проектов.[2]

Таким образом, прогнозируемые (инженерные) методологии фокусируются на детальном планировании будущего. Известны запланированные задачи и ресурсы на весь срок проекта. Команда с трудом реагирует на возможные изменения. План оптимизирован исходя из состава работ и существующих требований. Изменение требований может

привести к существенному изменению плана, следовательно, к изменению сроков и бюджета проекта.

Суммируя сказанное выше, необходимо сделать вывод о том, что инженерные методологии по определению были не способны разрешить кризис программного обеспечения. Новые и новые неудачи проектов, использующих инженерные методологии, подталкивали к поиску альтернативных подходов к разработке. И они не замедлили появиться.

По мере осознания проблем инженерных методологий стали появляться альтернативные подходы к разработке, призванные устранить их недостатки.

Во второй половине 90-х годов сформировалась и обрела популярность целая группа так называемых «гибких методологий». Несмотря на некоторые отличия в используемых практиках, эти подходы были схожи в том, что в качестве альтернативы инженерным методологиям они стремились предложить адаптивные, итерационные, ориентированные на человека подходы[2].

Гибкие методологии – это подходы к разработке программного обеспечения, ориентированные на использование итеративной разработки, динамическое формирование требований и обеспечение их реализации в результате постоянного взаимодействия внутри самоорганизующихся рабочих групп, состоящих из специалистов различного профиля[5].

Для построения жизненного цикла программного продукта используется инкрементная модель.

Инкрементная модель жизненного цикла процесса разработки ПО основана на разбиении всего процесса разработки на отдельные этапы (инкременты), каждый из которых включает основные этапы каскадной модели: анализ, проектирование, кодирование и тестирование.

На первом этапе реализуются базовые функции программного обеспечения, при этом процесс разработки проходит все этапы классического цикла. Выполняется системный анализ объекта автоматизации и анализ требований, строится план реализации первой версии ПО, выполняется проектирование, реализация и тестирование. В конце первого этапа разработчик предоставляет заказчику рабочий программный продукт с ограниченной функциональностью, проходит сдача-приемка работ. При этом желательно внедрение полученных результатов работы в опытную или даже промышленную эксплуатацию. Этот факт должен учитываться при выборе функций первой очереди. На вход второго и последующих этапов идет программный продукт и список требований по его доработке. Все этапы жизненного цикла повторяются. Таким образом, весь проект разбивается на множество мини-проектов, которыми легче управлять.

В рамках каждого инкремента возможно точное определение требований, сроков разработки, ресурсов и затрат. При этом на последующих инкрементах возможно исправление неточностей, допущенных при первоначальном формулировании требований.

Работоспособность программного продукта, получаемого в результате выполнения каждого этапа позволяет существенно сократить сроки внедрения в эксплуатацию отдельных частей и программного комплекса в целом, а также более точно оценить результаты работы по этапам и формировать задания на доработку[3].

Гибкие методологии нацелены на преодоление ожидаемой неполноты требований и их постоянного изменения. Когда меняются требования, команда разработчиков тоже меняются. Команда, участвующая в гибкой разработке, с трудом может предсказать будущее проекта. Существует точный план лишь на ближайшее время. Более удаленные во времени планы существуют лишь как декларации о целях проекта, ожидаемых затратах и результатах[1].

Одной из самых популярных гибких методологий является Scrum.

Scrum — это методология управления разработкой информационных систем, в которой делается жесткий акцент на качественном контроле процессом разработки. Помимо управления проектами по разработке программного обеспечения, методика используется командами поддержки программного обеспечения, а также как подход управления разработкой и сопровождением программ.

По методологии Scrum, весь процесс разработки делится на небольшие периоды, которые называют спринтами. В ходе спринта происходит функциональный рост разрабатываемого программного обеспечения. Дробление задач на спринты, позволяет делать процесс разработки предсказуемым и гибким.

Термины, которые важны в методологии Scrum – это бэклог проекта и бэклог спринта. Под бэклогом проекта понимается упорядоченный по степени важности список требований к функциональности, в бэклоге спринта содержится функциональность, выбранная владельцем проекта из резерва проекта. Соответственно, процессом разработки проекта занимается Scrum-команда, участники которой наделяются строго определенными «ролями».

Важнейшие фигуры Scrum-процесса: ScrumMaster (проводит совещания (Scrum meetings), следит за соблюдением принципов процесса и разрешает противоречия), Product Owner (представляет интересы конечных пользователей), и собственно Scrum-команда (состоящая из специалистов разных профилей команда разработчиков проекта).

В ходе работ над проектом в соответствии с методологей Scrum происходят регулярные встречи команды, жесткая проверка исполненных задач, постановка новых

целей, корректировка. Существует целый ряд средств управления проектами, которые поддерживают ведение Scrum-процесса.

Преимуществом использования этой методологии является увеличение скорости запуска проекта с самыми приоритетными функциями и существенная экономия бюджета заказчика. Помимо этого, методология Scrum позволяет вести постоянный контроль над ходом работ и более гибко контролировать траты из бюджета проекта[4].

От теоретико-методологических основ предлагается перейти к рассмотрению применения Scrum на примере студии мобильной разработки ООО «65 Гигабайт».

Деятельность ООО «65 Гигабайт» направлена на удовлетворение требований заказчиков в части разработки мобильных приложений. Мобильные приложения – это программы, установленные на смартфонах.

В своей деятельности управления проектами компания использовала прогнозируемую методологию, основанную на каскадной модели жизненного цикла программного продукта. В результате накопления негативного опыта использования данной методологии в виде срыва сроков и увеличения планируемого бюджета проектов было принято решение пересмотреть политику компании в отношении системы управления проектами.

Одной из задач компании стал выбор адекватных современным условиям (меняющимся в ходе разработки требования заказчиков к мобильному приложению) проектных методологий и инструментальных средств управления проектами по разработке мобильных приложений. Сложность задачи была обусловлена большим количеством инструментальных средств, сложными взаимосвязями между ними, а также непрерывным развитием теории управления разработки ПО.

Целью при решении данной задачи было обеспечение “эффективного” диалога между менеджментом, участниками проектных команд, клиентами и субподрядчиками.

Анализ модели бизнеса компании, ее стратегии, культуры компании, особенностей выполняемых сегодня проектов (технологии, риски, доступность финансовых ресурсов) определил выбор в пользу гибкой методологии Scrum, основными принципами которой являются:

приоритет в удовлетворении потребностей заказчика;

на каждой итерации могут изменяться требования;

частые демонстрации нового функционала, чтобы согласовать дальнейшие возможные изменения в требованиях;

тесная работа заказчиков и разработчиков;

систематический анализ возможных способов улучшения эффективности своей работы.

Процесс внедрения методологии в компании состоял из нескольких мероприятий:

директором был издан приказ о вступлении в силу Регламента ведения проектов по разработке программного обеспечения;

распределение полномочий выстраивается по матричному типу (т.е. приходится отказываться от функционального типа организации);

для формализации процессов разработки используется менеджер задач JIRA, который был куплен ранее и имеет всю требуемую функциональность, поэтому, отдельной задачи выбора подобного инструмента не стояло;

изменена финансовая система мотивации сотрудников компании;

в отдел персонала отправлен запрос на проведение мероприятий по формированию командного духа;

проведено общее собрание коллектива компании, на котором были разъяснены цели и задачи перехода к новому регламенту работы, а также выгоды, которые могут принести сотрудникам эти изменения в рабочем процессе.

Нововведения оказали положительный эффект на скорость и качество разрабатываемого программного обеспечения, мотивацию сотрудников, основные риски проектов по разработке программного обеспечения, появилась определенность в достижении результатов проектов.

Результатами, которые достигла компания, используя Scrum в течении года, явились:

прозрачность процесса, ежедневное отображение хода выполнения работ за счет внедрения корпоративной системы управления проектами с компонентом Scrum. В этой системе предусмотрено использование инструмента Scrum-board (доска движения задач, поставленных конкретным разработчикам, на которой в режиме реального времени отображаются статусы выполнения всех находящихся в разработке задач).

предсказуемость сдачи промежуточных и финального результатов. Поскольку длительность каждого спринта фиксирована, заказчик и исполнитель знают даты получения промежуточных результатов работ, что позволяет контролировать ход выполнения работ по проекту.

повышение качества продукта: лучшее соответствие ожиданиям пользователей, уменьшение количества ошибок, за счёт их раннего обнаружения. Заказчик включен в непосредственно сам процесс разработки, участвует в планировании спринтов, в приемке промежуточных результатов, вместе с командой разработчиков определяет приоритетность выполнения задач.

увеличение продуктивности за счёт полного использования потенциала командной работы и фокусировки на производительности команды, а не на индивидуальной продуктивности;

самоорганизация команды повысила мотивацию и обеспечила обратную связь для корректировки процесса, что значительно уменьшило нагрузку на менеджмент;

упрощение вхождения в команду новых игроков за счёт ясности процесса, общей процессной терминологии, а также создания почвы взаимного обучения в виде ретроспектив и стенд-апов (регулярных встреч и обсуждений).

Таким образом, применение гибкой методологии Scrum было целесообразным и помогло решить проблемы, которые возникали при использовании прогнозируемой методологии.

Список литературы

1. Колтунова Е.В. Выбор методов, моделей и стандартов управления разработкой программного обеспечения// Диссертационное исследование.- СПб.: Питер, 2007.- 184 с.

2.Разумовский К. Введение в гибкую разработку программного обеспечения/ К. Разумовский [Электронный ресурс]. – Режим доступа: http://www.kv.by/index2008334201.htm

3.Технологии программирования: учебное пособие / С. А. Мытник.- Томск: ТУСУР, 2011.- 196 с.

4.Fowler Martin The new methodology / Martin Fowler [Электронный ресурс]. - Режим доступа: http://martinfowler.com/articles/newMethodology.html

5.Manifesto for Agile Software Development [Электронный ресурс]. – Режим доступа: http://agilemanifesto.org/

Рецензенты:

Осипов А.К., д.э.н., профессор кафедры «Менеджмент», ФГБОУ ВПО «Ижевский Государственный Технический Университет имени М.Т. Калашникова», г. Ижевск;

Соколова Н.Г., д.э.н., профессор кафедры «Экономика, технология и управление коммерческой деятельностью», ФГБОУ ВПО «Ижевский Государственный Технический Университет имени М.Т. Калашникова», г. Ижевск.