Правительство Российской Федерации

Федеральное государственное автономное   
Образовательное учреждение  
высшего профессионального образования

«Национальный исследовательский университет   
«Высшая школа экономики».

Факультет бизнес-информатики

Кафедра бизнес-аналитики

**ВЫПУСКНАЯ КВАЛИФИКАЦИОННАЯ РАБОТА**

На тему

«Управление проектом внедрения специализированного программного обеспечения в Банк»

Студент группы № 471  
   
Мазурина Наталья Алексеевна

Научный руководитель  
  
Богданова Татьяна Кирилловна,  
 доцент, к.э.н.

Рецензент  
  
Ямпольский Сергей Михайлович  
доцент

Москва 2013

**Оглавление**

[Введение 3](#_Toc357263855)

[Глава I. Характеристики и особенности управления IT-проектом 6](#_Toc357263856)

[1.1 Определение проекта 6](#_Toc357263857)

[1.2. Понятие управления проектами и команда проекта 8](#_Toc357263858)

[1.3 Управление IT проектом 10](#_Toc357263859)

[1.3.1. Определение IT проекта и его особенности 10](#_Toc357263860)

[1.3.2. Модели жизненного цикла IT проекта: их характеристики и особенности 11](#_Toc357263861)

[1.4. Постановка проблемы 18](#_Toc357263862)

[1.4.1 Специфика банковской сферы 18](#_Toc357263863)

[1.4.2 Анализ применимости моделей жизненного цикла в банковской сфере 19](#_Toc357263864)

[Глава II. «Гибкие» методологии управления IT-проектами 22](#_Toc357263865)

[2.1. Описание и принципы «гибких» методологий управления IT-проектами 22](#_Toc357263866)

[2.2. Обоснование выбора методологии «скрам» для моделирования процесса управления IT-проектом 26](#_Toc357263867)

[2.3. Описание методологии «скрам» 30](#_Toc357263868)

[Глава III. Управление проектом внедрения специализированного программного обеспечения в Банк 40](#_Toc357263869)

[3.1. Характеристика внедряемого программного обеспечения в Банк 40](#_Toc357263870)

[3.2. Управление проектом внедрения ПО на основе водопадной модели 41](#_Toc357263871)

[3.3. Моделирование и анализ внедрения проекта с помощью «гибкой» методологии «скрам» 50](#_Toc357263872)

[Заключение 61](#_Toc357263873)

[Список использованной литературы 64](#_Toc357263874)

[ПРИЛОЖЕНИЕ 68](#_Toc357263875)

# Введение

Управление проектами сегодня отличается от управления проектами в 1950-х, когда эта наука стала впервые применяться строительными и инженерными копаниями в качестве инструмента для эффективной организации новой деятельности. Сейчас управление проектами используется практически во всех областях, так как это позволяет поддерживать экономическую активность организаций.

С развитием информационных технологий многим компаниям стало очевидно, что, используя различные автоматизированные информационные системы, можно повысить рентабельность бизнеса. Особенно это стало актуально для российских банков, так как они пытаются повысить свою конкурентоспособность на рынке с помощью использования специализированных программных систем, ориентированных на банковскую деятельность. Таким образом, с повышением степени вовлеченности IT-продуктов в жизнь общества, тема об эффективном управлении IT-проектами стала очень популярной.

В данной дипломной работе управление IT-проектом будет рассматриваться в банковской сфере, которая считается довольно уязвимой из-за сильной коррелированности с внешней средой и экономикой страны. Ввиду этих специфик, а также характеристик IT-проекта, управление внедрением или созданием ПО в банках сталкивается с множеством трудностей, такими как сдвиги по срокам, ненадлежащее качество, превышение по запланированной стоимости и тому подобное. Ненадлежащее качество продукта, являющегося результатом управления какого-либо IT-проекта в банке, может сильно сказаться на дальнейшей экономической активности этого банка.

Во многих российских банках жизненный цикл IT-проектов рассматривается с точки зрения традиционной водопадной модели (то есть в виде модели, где каждый последующий этап процесса начинается только тогда, когда завершится предыдущий), применение которой основано, скорее, на привычке менеджеров проектов, нежели на реальной эффективности модели. В последнее время большую популярность приобретают «гибкие» методологии управления IT-проектами, основная суть которых базируется на преимуществах итеративной и спиральной моделей жизненного цикла. Помимо этого, «гибкие» методологии пропагандируются как «готовые к изменениям», что в большинстве случаев больше подходит для IT-проектов, управляемых в быстро меняющихся средах. **Этим объясняется актуальность выбранной темы для исследования.**

**Объектом исследования** является IT-проект по внедрению программного обеспечения в банк. **Предметом исследования** являются способы управления IT-проектом, основанные на том или ином представлении жизненного цикла этого IT-проекта (модели и методологии).

Таким образом, **целью данной работы** является выявление целесообразности применения «гибкой» методологии «скрам» для управления IT-проектом вместо традиционного «водопадного» на примере IT-проекта по внедрению специализированного программного обеспечения для автоматизированной оценки кредитоспособности клиента и для принятия решения о выдаче кредита в Банк «XXX».  Критерием оценки целесообразности является реализация проекта на заданном уровне качества при значительно меньшем уровне затрат и времени.

**В рамках поставленной цели были определены следующие задачи**:

* выявление характеристик проекта и управления проектом;
* определение критериев успешности проекта;
* определение особенностей IT-проекта;
* анализ преимуществ и недостатков моделей жизненного цикла управления IT-проектом;
* определение сущности и преимуществ «гибких» методологий управления IT-проектами;
* анализ различных видов «гибких» методологий и обоснование выбора методологии «скрам» как наиболее подходящей для исследования;
* описание основных моментов, характеристик и особенностей методологии «скрам»;
* реализация проекта по внедрению программного обеспечения в Банк с использованием традиционной водопадной модели жизненного цикла;
* предложение по управлению этим же IT-проектом с использованием «гибкой» методологии «скрам»;
* анализ и сравнение результатов с учетом определенных критериев успешности IT-проекта.

Дипломная работа состоит из трех глав, введения, заключения, списка использованной литературы и приложения.

Первая глава посвящена понятиям «проект», «управление проектом», «IT-проект», «жизненный цикл IT-проекта», а также анализу моделей жизненного цикла IT-проекта. В конце главы рассматриваются характеристики банковской сферы и применимость описанных моделей жизненного цикла в зависимости от этих характеристик.

Во второй главе рассматриваются «гибкие» методологии управления IT-проектами, проводится их сравнительный анализ и выбирается методология «скрам» для моделирования и управления конкретным IT-проектом в практической части.

В третьей главе реализуется проект внедрения специализированного программного обеспечения для автоматизированной оценки кредитоспособности клиента и для принятия решения о выдаче кредита, и моделируется этот же проект с использованием методологии «скрам».

# Глава I. Характеристики и особенности управления IT-проектом

## 1.1 Определение проекта

Термин «проект» берет свои истоки в латинском языке, происходя от слова «projecere» - бросаться вперед. В английском языке слово «project» сохранило базовую суть из латинского языка, сузив, при этом, основное значение – деятельность, направленная на достижение поставленной цели. В современном обществе операционное управление бизнесом, которое нацелено на управление устоявшимися бизнес-процессами, теряет свою актуальность. Это связано в первую очередь с тем, что внешняя среда постоянно претерпевает резкие и существенные изменения, соответственно, большую популярность приобретают проекты, то есть такие активности, которые характеризуются определенной степенью уникальности при решении какой-либо задачи. Помимо всего прочего, у организации может быть множество целей, следовательно, появляется огромное число проектов, разных по своей природе, и от их управления может зависеть будущее всей организации. Для того чтобы какая-либо активность смогла называться проектом, необходимо, чтобы у нее одновременно присутствовали нижеприведенные характеристики:

* Четко-определенная цель (как правило, это получение какого-то конечного результата, удовлетворяющего организацию);
* Установленная продолжительность проекта с ранее определенными датами начала и завершения проекта;
* Предопределенные требования к ресурсам (время, оборудование, финансы и труд);
* Координированное участие различных специалистов в одном проекте (то есть, синтез многочисленных разнонаправленных действий)

В современных книгах все определения термина «проект» базируются на этих характеристиках. Таким образом, предлагаются следующие определения:

Проект - это

1. «комплексное, неповторяющееся, единовременное мероприятие, ограниченное по времени, бюджету, ресурсам, а также четкими указаниями по выполнению, разработанными под потребности заказчика» [4, с. 13];
2. «системный комплекс плановых (финансовых, технологических, организационных и прочих) документов, содержащих комплексно-системную модель действий, направленных на достижение оригинальной цели» [11, с. 21].

Проекту так же свойственна зависимость между четырьмя показателями: бюджет (заданные в плане оценочные затраты на проект), длительность (период времени, необходимый для выполнения проекта), область охвата (цели и задачи проекта) и качество (рисунок 1):



Рис. 1 Треугольник проекта

Изменение одного из этих элементов влияет на три других. Ограниченность бюджета определяется количеством средств, которое было выделено для реализации проекта. Ограниченность длительности определяется временным периодом, в рамках которого проект должен быть реализован. Ограниченность области охвата определяется действиями, которые должны быть выполнены для того, чтобы цель проекта была достигнута. Сбалансированность этих показателей может свидетельствовать об успешности проекта.

## 1.2. Понятие управления проектами и команда проекта

До XX века проектами управляли непосредственно их создатели, а ими, как правило, являлись архитекторы, инженеры и строители, однако вскоре и другие организации начали осознавать, что грамотное управление проектом является очень важным для их деятельности. Особенно отчетливо это понималось в сферах строительства, консалтинга и обороны. Со временем стало увеличиваться количество людей, стремящихся к познанию в этой области: если несколько лет назад в университетах можно было встретить курсы по управлению проектами, которые читались исключительно для студентов, обучающихся на инженерных специальностях, то сейчас напротив, все больше и больше факультетов включают дисциплину «управление проектами» в свои программы обучения. Таким образом, выпускники факультетов маркетинга, информационных технологий, экономики, финансов, медицины и других наук имеют представление об управлении проектами, что, безусловно, важно в сегодняшние дни, потому что управление проектами охватывает не только сферы бизнеса и строительства, как раньше, но и любые стороны деятельности общества.

В современной литературе термин «управление проектами» подразумевает под собой различные методы, способы и идеи о том, как сделать бизнес более эффективным. В качестве примера ниже приводятся определения, описанные в различных российских и зарубежных источниках:

Управление проектами - это

1. *«*методология (говорят также — искусство) организации, планирования, руководства, координации трудовых, финансовых и материально-технических ресурсов на протяжении проектного цикла, направленная на эффективное достижение его целей путем применения современных методов, техники и технологии управления для достижения определенных в проекте результатов по составу и объему работ, стоимости, времени, качеству и удовлетворению участников проекта» [9, с. 18];
2. «область менеджмента, охватывающая те сферы производственной деятельности, в которых создание продукта или услуги реализуется как уникальный комплекс взаимосвязанных целенаправленных мероприятий при определенных требованиях к срокам, бюджету и характеристиками ожидаемого результата» [12, с. 15];
3. «профессиональная деятельность по руководству ресурсами (человеческими и материальными) путем применения методов, средств и управления для успешного достижения заранее поставленных целей в результате выполнения комплекса взаимосвязанных мероприятий при определенных требованиях к срокам, бюджету и характеристикам ожидаемых результатов проектов» [6, с. 7];
4. «особый вид управленческой деятельности, базирующийся на предварительной коллегиальной разработке комплексно-системной модели действий по достижению оригинальной цели и направленный на реализацию этой модели» [11, с. 22].

В рамках управления проектом происходит постоянное взаимодействие участников проекта, к которым относятся инициатор, заказчик, менеджер проекта, инвестор и, непосредственно, команда проекта.

Основная идея проекта устанавливается инициатором. В дальнейшем, ее перенимает заказчик, который заинтересован в определенных результатах проекта. Заказчиком определяются желаемые временные рамки проекта, а также различные требования к проекту. Финансирование проекта осуществляется либо заказчиком, либо инвестором, который может вступать в контрактные отношения с заказчиком. Менеджер проекта занимается управлением деятельностью, направленной на реализацию поставленных целей в рамках проекта. Он несет прямую ответственность за достижение целей и за возможные ошибки перед заказчиком. Под руководством менеджера проекта происходит формирование команды проекта в зависимости от сложности организации проекта и возложенных на команду обязательств. При наборе людей в команду менеджер проекта должен учитывать их профессиональную квалификацию и опыт работы. Таким образом, чем сложнее является проект, тем более квалифицированной должна быть команда.

## 1.3 Управление IT проектом

### 1.3.1. Определение IT проекта и его особенности

С развитием информационных технологий одна часть организаций стала автоматизировать свою деятельность, а вторая часть начала создавать программное обеспечение для его дальнейшей поставки организациям первого типа. И в первом и во втором случае, организации имеют дело с IT-проектами.

Термин «IT-проект» можно описать в виде общего термина «проект» но с некоторыми важными дополнениями, связанными, прежде всего, с тем, что IT-проект подразумевает собой деятельность, направленную на создание или использование информационных технологий (Грекул, Коровкина, 2011). Так как в данной работе пойдет речь об управлении проектом внедрения специализированного программного обеспечения (то есть, некой информационной технологии), то целесообразно определить, что именно отличает IT-проект от обычного проекта. Во-первых, диссонанс между заказчиком и исполнителем. Чаще всего IT-проекты заказываются бизнес-отделом организации, а внедряются – собственным IT отделом или самостоятельной IT-компанией (отдаются на аутсорсинг). При таком подходе очевидно, что возможны трудности в определении бизнес-требований и общих ожиданий от разрабатываемого и внедряемого программного обеспечения, так как область IT является довольно специфичной. Это означает, что на первых этапах проводимые обсуждения между заказчиком и исполнителем могут нести в себе достаточно размытые идеи, которые на ранних стадиях фиксируются наспех в техническом задании, а потом меняются. При первичном чтении технического задания у заказчика может возникнуть ряд вопросов, и это нормально, потому что разработчик технического задания мог понять какие-то аспекты IT-проекта очень субъективно. Другой особенностью IT-проекта является равномерное распределение вины на участниках проекта при его неблагополучной реализации. Таким образом, ответственность за результат ложится не только на исполнителя (за возможную неправильную реализацию) или не только на заказчика (за возможную ошибку в формировании требований), а на всех участников вместе, потому что, так или иначе, они должны создавать между собой эффективную коммуникацию, минимизируя возможный субъективизм в дальнейшем. Третьей особенностей IT-проекта можно назвать его высокую стоимость. Если при построении здания возможны лишь минимальные отклонения от конечных требований и ожиданий, то в IT-проектах это весьма вероятно. Соответственно, есть риск внесения существенных изменений, а так как очень часто отношения между заказчиком и исполнителем регулируются ранее подписанным контрактом, то любое изменение влечет за собой так называемый «запрос на изменение», который стоит определенную сумму денег. Большое число изменений подразумевает под собой и большую стоимость, именно поэтому IT-проекты считаются одними из самых дорогих видов проектов.

### 1.3.2. Модели жизненного цикла IT проекта: их характеристики и особенности

#### 1.3.2.1. Определение жизненного цикла IT-проекта

Каждый IT-проект проходит путь с момента своего создания через ряд промежуточных этапов и до его завершения, когда информационная технология полностью создана или внедрена. Другими словами можно сказать, что у проекта есть фазы, которые называются общим термином - жизненный цикл. Свод знаний по управлению проектами (PMBoK) дает следующее определение понятию жизненный цикл проекта:

«Жизненный цикл проекта – это набор, как правило, последовательных и иногда перекрывающихся фаз проекта, названия и количество которых определяются потребностями в управлении и контроле организации или организаций, вовлеченных в проект, характером самого проекта и его прикладной областью» [27, с. 15].

Жизненный цикл помогает координировать усилия участников проекта, так как он учитывает некоторую специфику. К примеру, руководитель проекта должен понимать, какое влияние оказывают входы и выходы одного этапа жизненного цикла на остальные. Жизненный цикл проекта имеет прямую связь с его финансированием, так как, только спланировав фазы его реализации, можно прогнозировать стоимость и издержки.

В наиболее общих чертах схема жизненного цикла IT-проекта может быть представлена четырьмя фазами: инициация, планирование, выполнение и закрытие (рисунок 2):

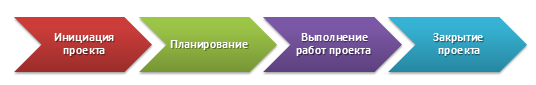


Рис. 2 Схема жизненного цикла IT-проекта

* Инициация: создается идея проекта, определяются бизнес-требования и цели;
* Планирование: составляются план проекта, финансовый план, план качества и план ресурсов; определяется работа, которая должна быть сделана в рамках проекта;
* Выполнение: исполнение задач; полученные результаты оцениваются руководителем проекта и впоследствии сравниваются с планом проекта;
* Закрытие: окончательный результат и сформированная документация представляются заказчику; формируется уровень успеха проекта.

#### 1.3.2.2. Водопадная модель жизненного цикла IT-проекта

Водопадная модель была введена примерно в 1970 году, и она является традиционным подходом к управлению IT-проектами. Водопадный процесс представляет собой упорядоченную последовательность различных фаз развития проекта, причем каждая фаза имеет набор входов и выходов (Holtsnider, 2010).

Основные фазы водопадной модели жизненного цикла IT-проекта представлены на рисунке 3.

* «Разработка требований». Происходит сбор бизнес-требований заказчика, после чего осуществляется их преобразование в функциональные требования к программному продукту;
* «Архитектура и дизайн». Определение требований, относящихся к программному обеспечению (базы данных, параметры безопасности, модели данных и интерфейсная схема);
* «Разработка». Создание программного продукта, базирующегося на спецификациях, определенных на предыдущих стадиях;
* «Тестирование». Проверка отсутствия в программном продукте каких-либо дефектов, а также соответствия функциональности требованиям заказчика
* «Внедрение». Инсталляция продукта;
* «Поддержка». Проверка корректности в дальнейшей работе программы.

(Ledbrook, 2012), (Melonfire, 2006).

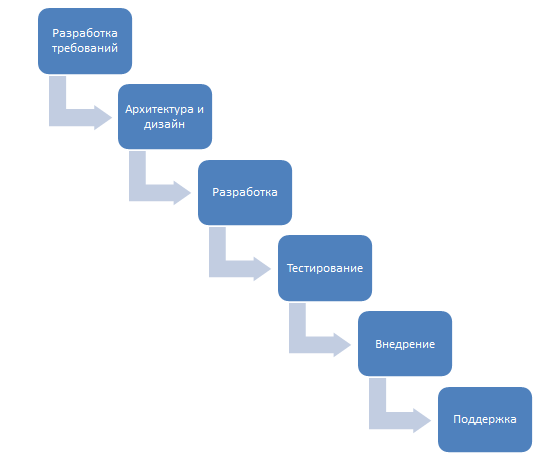


Рис. 3 Водопадная модель жизненного цикла IT-проекта

Ключевым моментом водопадной модели является ее последовательный характер, что означает, что каждая фаза не может начаться, если не завершена предыдущая. Данная модель требует четкого планирования в самом начале проекта, и это всегда сопровождается написанием больших технических заданий, а результат работающей программы можно наблюдать только в конце. Управлением IT-проектом в «водопаде» руководит менеджер проекта, который поручает задания разработчикам и устанавливает сроки и бюджет. На протяжении всего жизненного цикла команда проекта отчитывается по проделанной работе только перед менеджером, коммуникации непосредственно с заказчиком отсутствуют. Если определенные в начале проекта требования не меняются, то руководствоваться техническим заданием весьма целесообразно. Проблема заключается в том, что на деле ни один IT-проект нереально спроектировать таким образом, чтобы заранее предугадать все требования пользователей, его функционал и так далее. Помимо всего прочего, если что-то было упущено в работе проекта на его начальных стадиях, исправить это, находясь на завершающих этапах, очень сложно и требует дополнительных затрат.

Подводя итог, можно отметить, что преимущества данной модели имеют место только в небольших проектах при четко определенных и неизменяемых требованиях; в других же случаях, скорректированные планы могут способствовать невыполнению проекта.

#### 1.3.2.3. Итеративная модель жизненного цикла IT-проекта

Итеративная модель жизненного цикла значительно отличается от водопадной. Данный подход подразумевает последовательность циклических итераций, где каждая из них состоит из четырех фаз: сбор требований, планирование, разработка и тестирование (рисунок 4) (Jalote, 2004).

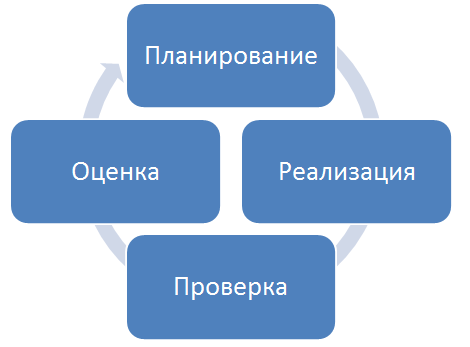


Рис. 4 Фрагмент итерации итеративной модели жизненного цикла IT-проекта

Главной характеристикой итеративной модели является то, что каждая итерация добавляет новый функционал в программный продукт. Для Jalote [32, с. 117] «новая итерация начинается до завершения текущей, тем самым, являясь продолжением актуальной версии программного обеспечения». В этом случае, направление проектной деятельности может быть изменено в начале каждой итерации, что является довольно удобным в случае быстро меняющейся внешней среды. Функциональность всей системы может быть глобально оценена уже в начале проекта (на первых итерациях), а необходимые второстепенные изменения могут быть сделаны в течение следующих итераций. Более того, итерации могут служить средством обратной связи для измерения качества выполнения проекта и продуктивности работы команды участников. Итеративный подход к планированию жизненного цикла предполагает, что различные виды работ над проектом не привязаны к каким-то конкретным стадиям разработки; вместо этого они выполняются по мере необходимости.

#### 1.3.2.4. Спиральная модель жизненного цикла IT-проекта

Развитием идеи итеративной разработки программного обеспечения является спиральная модель жизненного цикла, предложенная Барри Боэмом в 1986 году. Основной сутью данной модели является минимизация рисков в начале каждой итерации путем повышения коммуникации внутри команды проекта. В начале каждой итерации спиральной модели разработчики идентифицируют:

* Цели разрабатываемого в течение данной итерации фрагмента программного обеспечения;
* Основные и альтернативы пути к достижению поставленных целей;
* Анализ возможных ограничений.

После этого команда проекта начинает идентифицировать проблемы, которые могут возникнуть в течение итерации, а также причины их возникновения (нехватка информации, недостаточная квалификация сотрудников). Далее команда начинает разрабатывать стратегию осуществления проектных работ в рамках данной итерации с учетом обнаруженных рисков.

Каждый «виток спирали» представляет собой законченную часть разрабатываемого, продукта, соответственно, с каждым новым «витком» проект достигает более глубокий уровень детализации и конкретизации (рисунок 5):

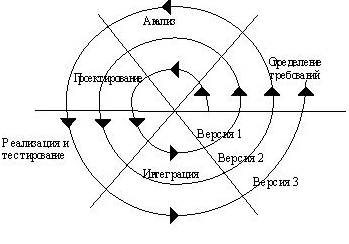


Рис. 5 Спиральная модель жизненного цикла IT-проекта

Планированию каждой итерации-«витка» уделяется особое внимание, так как Боэм полагал, что неудачная реализация итераций связана, прежде всего, с низкой коммуникацией специалистов в команде проекта. Одним из явных недостатков спиральной модели является сложность определения момента для перехода на следующую фазу (Грекул, 2010).

С развитием теории управления IT-проектами разработчики и их менеджеры поняли, что объединенный вариант итеративной и спиральной моделей жизненного цикла может стать лучшим способом для организации процесса управления IT-проектом. Пошаговая разработка и наличие готовых фрагментов работающего ПО, взятые из итеративной модели, а также главенствующая роль человеческого фактора и анализ возможных рисков, взятые из спиральной модели были объединены в новую методологию, получившую название **«гибкой» методологии** (agile – англ). Под гибкостью подразумевается следующее:

* Возможность быстро реагировать на изменения для того, чтобы преуспеть в беспокойной бизнес-среде;
* Возможность быстрого изменения степени приоритета используемых ресурсов в ответ на изменения в требованиях, технологиях и знаниях;
* Возможность быстрой реакции на любые угрозы рынка, а также на любые изменения, вызванные влиянием заказчиков;
* Использование инкрементального подхода в поставке продукта для максимального удовлетворения требованиям заказчика;
* Максимальное увеличение рентабельности проекта путем стремления завершить все работы в срок.

(Cobb, 2011).[[1]](#footnote-1)

## 1.4. Постановка проблемы

### 1.4.1 Специфика банковской сферы

Сейчас в России насчитывается 939 банков[[2]](#footnote-2), причем довольно велико и количество стартапов. Конкуренция начинает возрастать, и банкам приходится искать новые пути к завоеванию больших объемов рынка. Рынок требует от банков высокой адаптивности к возможным изменениям внешней среды, а адаптивному ведению бизнеса способствуют информационные технологии, поэтому одним из главных показателей успешности банка в настоящий момент является степень его автоматизации.

Идеи об организации автоматизации банковских процессов начали появляться недавно, и это связано с возросшей тенденцией выпускать различные внешние автоматизированные сервисы, предоставляющие определенную информацию о клиентах. Если взять во внимание рынок кредитования, то в качестве примеров таких сервисов можно, безусловно, привести программный комплекс CreditRegistry, выпущенный компанией ЗАО «МТЦ». CreditRegistry автоматизирует бизнес-процессы, связанные со взаимодействием банка с бюро кредитных историй и другими сервисами, помогающими банку организовать свою работу на рынке потребительского кредитования[[3]](#footnote-3). Развитие таких сервисов стимулирует банки автоматизировать работу внутри своих департаментов, получая значительную часть данных о клиентах извне.

Банковская сфера является одной из быстро меняющихся и рискованных, что имеет прямое влияние на управление проектами в банках. Достаточно вспомнить кризис 2008 года, после которого множество банковских проектов было попросту заморожено. Банковская сфера также связана с экономической деятельностью государства, соответственно, любые изменения в экономике страны способны изменить актуальность каких-либо данных, использующихся в банках. Таким образом, те IT-проекты, которые предназначаются для автоматизации именно специфических банковских функций, с большей вероятностью будут подвержены изменениям из-за изменений окружающей среды в течение периода их реализации. Банкам также характерны многочисленные конфликты между департаментами, и это зависит от различий в видениях бизнеса. К примеру, рассматриваемый в данной работе проект по внедрению программного обеспечения по принятию решения о выдаче кредита является синтезом идей трех банковских департаментов: рисков, операционного отдела и отдела продаж и маркетинга. Каждый департамент формирует требования, отсортированные по приоритету, руководствуясь собственными мотивами: для отдела продаж главным является повышение количества выданных кредитов (что способствует дальнейшему получению прибыли), для операционного отдела преимуществом является удобность и лаконичность в пользовании программой, риски же стремятся минимизировать вероятность невыплат по кредитам, отбирая, тем самым, только «нерискованных» клиентов.

### 1.4.2 Анализ применимости моделей жизненного цикла в банковской сфере

Суммируя информацию, характеризующую банковскую сферу, можно отметить, что внедрение IT-проекта в банк с большей долей вероятности будет претерпевать постоянные изменения в требованиях к функционалу программного обеспечения. Стоит также добавить, что для заказчика (в данном случае, для Банка) внедряемое ПО является инструментом для исполнения функций, соответствующих характеру его бизнеса, а так как конфликты между отделами банка встречаются довольно часто, то и к общему соглашению отделы будут приходить постепенно, постоянно меняя решения.

Проанализировав специфику банковской сферы, а также особенности IT-проектов, можно предположить, что применение водопадной модели в данном контексте не является целесообразным решением.

При водопадной модели каждое изменение или неточность какого-либо требования принуждают команду проекта возвращаться на более раннюю фазу и повторять проделанную работу заново. Эта переработка выбивает проектную команду из графика и нередко приводит к росту затрат. Несмотря на то, что команда проекта может постараться подготовить грамотное техническое задание, демонстрирующее, на их взгляд, каждый аспект разрабатываемого программного продукта, вероятность обнаружения расхождений в дальнейшем все же велика, причем эти расхождения могут быть заметны только на этапе тестирования.

Как известно, стоимость исправления ошибок в проекте прямо пропорциональна его длительности. Таким образом, ошибки в требованиях, обнаруженные на этапе тестирования, могут быть катастрофическими как для финансовой стороны компании, так и для всего проекта в целом. Таким образом, применение водопадной модели жизненного цикла может спровоцировать появление проблем, характерных IT-проектам в быстро меняющихся средах.

Стоит обратить внимание на применение «гибкой» методологии управления этим IT-проектом, объединяющей в себе преимущества итеративной и спиральной моделей жизненного цикла. В условиях быстро меняющихся требований и нестабильной окружающей среды подход, который предлагает внедрять или разрабатывать программное обеспечение по частям с постоянным отслеживанием готового продукта и с постоянной коммуникацией между заказчиком и разработчиками, может быть намного результативнее, чем последовательное внедрение программы с большими объемами документации, что предлагает традиционный водопадный подход.

# Глава II. «Гибкие» методологии управления IT-проектами

## 2.1. Описание и принципы «гибких» методологий управления IT-проектами

Как было упомянуто ранее, теория «гибких» методологий появилась после осознания преимуществ итеративной и спиральной моделей жизненного цикла по сравнению с традиционной водопадной моделью. В феврале 2001 года состоялась встреча создателей методологии сочетающих в себе основные принципы итеративной и спиральной модели. На этой встрече было решено назвать такие методологии общим словом «гибкие» (agile). «Гибкая» методология – это итеративный процесс, использующий уникальные практики для получения нового функционала программного обеспечения каждые 1-4 недели (Rubio, 2008).

Тогда же в 2001 году были созданы два основополагающих документа: «Манифест гибких методологий разработки» и «Принципы гибкой разработки». В первом из них четко прописаны высокоуровневые ценности, которым уделяется внимание в процессе управления IT-проектом:

*«Люди и взаимодействие важнее процессов и инструментов  
Работающий продукт важнее исчерпывающей документации  
Сотрудничество с заказчиком важнее согласования условий контракта  
Готовность к изменениям важнее следования первоначальному плану»* [39].

Первая ценность говорит о том, что вне зависимости от того, какими бы важными процессы и инструменты ни были, успех проекта зависит, в первую очередь, от вовлеченных в этот проект людей, т.е. в «гибких» методологиях очень важным является человеческий фактор, а именно возможность организации эффективной коммуникации, легкость преодоления конфликтов и так далее. Вторая ценность вовсе не исключает документацию, наоборот, она упрощает возможность взаимодействия и сотрудничества, а также налаживает процесс передачи знаний, однако эффективное взаимодействие между людьми намного лучше сказывается на проекте, нежели хорошо описанная документация без взаимодействий внутри команды. Третья ценность подразумевает, что непрерывное общение между заказчиком и командой разработчиков является более целесообразной стратегией, нежели «война» сторон за правильную трактовку ранее подписанного контракта. «Современная практика реализации проектов, как правило, сопровождается затяжными переговорами между участниками. Разумеется, переговоры – один из важнейших элементов деловой практики. Но зачастую они превращаются в игру с нулевой суммой: одна из сторон пытается обеспечить собственные интересы за счет другой стороны» [14, с. 88]. «Гибкие» методологии противопоставляют такой «борьбе» диалог между разработчиками и заказчиками. Такой подход позволяет выигрывать двум сторонам общими усилиями. Последняя четвертая ценность призывает команду проекта быть постоянно готовыми к изменениям. Ни один IT-проект не может заранее быть спроектирован целиком и полностью, со всеми учтенными нюансами. «Гибкие» методологии не отрицают необходимость досрочного планирования, они лишь допускают внесение корректировок в течение времени, отведенного для реализации IT-проекта.

В «Принципах гибкой разработки» детализируются основные ценности из «Манифеста»[[4]](#footnote-4). Эти двенадцать конкретизированных ценностей являются синтезом основополагающих принципов итеративной и спиральной моделей. Они позволяют организовать дисциплинированный гибкий процесс управления IT-проектом, проводя всю работу итерациями с промежуточными проверками, создавая слаженную самоорганизующуюся команду разработчиков и осуществляя постоянную коммуникацию с заказчиком. («частые поставки продукта», «частая поставка новых версий программного обеспечения», «методы взаимодействия и обмена информацией», «техническое совершенство и качественная архитектура»). Важно отметить, что именно налаженное взаимодействие между участниками проекта способно гарантировать быстроту исполнения.

Гэри Чин в своей книге «Agile Project Management: How to Succeed in the Face of Changing Project Requirements» (Chin, 2004) более детально перечисляет факторы проекта, наличие которых должно служить сигналом для использования именно «гибкой» методологии управления IT-проектом. Данные факторы автором разделяются на две группы: «внутренние неопределенности» и «внешние неопределенности» (рисунок 6):



Рис. 6 Внутренние и внешние неопределенности проекта

Если проект подвержен каким-то внутренним или внешним неопределенностям, то вполне благоразумно будет планировать его жизненный цикл, используя «гибкие» методологии.

При использовании «гибких» методологий задачи проекта разбиваются на малые части (итерации) с тщательным краткосрочным планированием и почти незначительным долгосрочным планированием. Если в классической итеративной модели длина итерации может быть сколь угодно большой (но в разумных пределах), то «гибкие» методологии ограничивают время, отведенное на выполнение конкретной итерации, до пяти недель.

«Гибкие» методологии базируются на эмпирическом управлении, то есть на таком управлении, где решения принимаются, исходя из промежуточных результатов проекта. Кроме того, данные результаты являются прозрачными, что означает, что все вовлеченные в проект люди осведомлены статусом работ по проекту, количеством изменений и возможными проблемами. (Layton, 2012). Нельзя не отметить, что в случае IT-проектов эмпирическое управление позволяет быстро вносить корректировки в программный продукт, и это является большим преимуществом по сравнению с водопадной моделью жизненного цикла.

Последние несколько лет в США проводилось много исследований, направленных на изучение преимуществ «гибких» методологий по сравнению с традиционной водопадной моделью. Например, согласно седьмому ежегодному государственному исследованию «гибкой» разработки Интернет-ресурса Version One, 84% респондентов (разработчиков) признали, что «гибкие» подходы кажутся им более эффективными, чем традиционные подходы.[[5]](#footnote-5)

Кроме того, 70% респондентов этого же исследования признали, что IT-проекты, внедряемые с помощью «гибких» методологий, реализовываются намного **быстрее** IT-проектов, внедряемых с помощью водопадной модели жизненного цикла (рисунок 7):

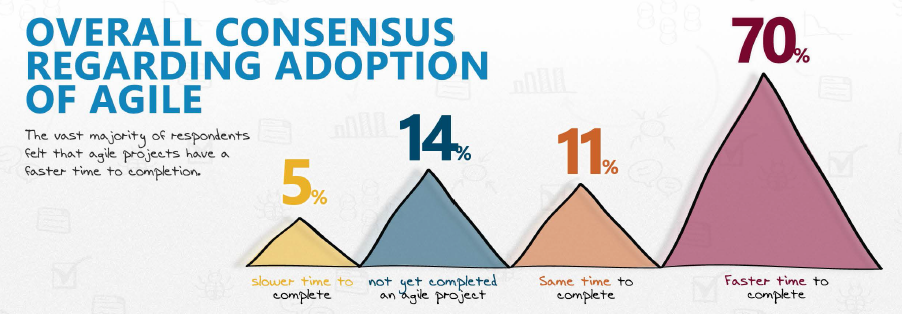


Рис. 7 Проекты внедряются быстрее, если используется "гибкая" методология

В качестве завершающего этапа анализа природы «гибких» методологий и их преимуществ ниже приведена таблица, в которой еще раз указаны основные различия «гибких» методологий и водопадной модели, которые не были отражены явным образом в «Манифесте»:

Таблица 1 Ключевые различия «гибких» методологий и водопадной модели

|  | **«Гибкие» методологии** | **Водопадная модель** |
| --- | --- | --- |
| **Требования заказчика** | Итеративный сбор | Детализированные требования четко определяются до начала фазы непосредственного внедрения |
| **Стоимость доработок** | Низкая | Высокая |
| **Направление разработки** | Может быть изменено в любой момент | Фиксированное |
| **Тестирование** | После каждой итерации | Во время соответствующей фазы, следующей за внедрением/разработкой |
| **Вовлеченность заказчика** | Высокая | Низкая |

## 2.2. Обоснование выбора методологии «скрам» для моделирования процесса управления IT-проектом

В настоящее время существует множество гибких методологий, но наиболее популярными являются «экстремальное программирование» (XP), «скрам», «бережливая разработка» (lean), «разработка, управляемая функциональностью» (fdd) и «канбан». В таблице 2 приведен сравнительный анализ данных методологий (Abrahamsson, 2002):

Таблица 2 Сравнительный анализ "гибких" методологий

| **Название** | **Ключевые моменты** | **Уникальные особенности** | **Недостатки** |
| --- | --- | --- | --- |
| «экстремальное программирование» (XP) | Маленькие команды, ежедневные совещания, неформальный тип коммуникации внутри команды, минимум документации | Постоянное корректирование проекта для повышения его эффективности и для адаптации к изменениям | Больше подходит для индивидуальных практик, чем для глобального управления, так как в последнем случае есть риск формирования недисциплинированных команд |
| «скрам» | Независимая, небольшая, самоорганизующаяся команда разработчиков, длина итерации – 2-4 недели, команда сама решает сколько времени ей нужно для выполнения задачи, неформальный тип коммуникации внутри команды, ежедневные совещания, присутствует только базовая документация | Высокий уровень коммуникации и взаимодействия внутри команды, четко прописанная формальная организация этой методологии, наличие «надсмоторщика» за командой, полная ориентация на требования заказчика | Есть риск увеличения времени проекта за счет издержек «скрам»-мероприятий |
| «бережливая разработка» (lean) | Использование визуализирующих инструментов, разработка через тестирование, короткие итерации | Главными являются те функции ПО, которые **ценны** для заказчика, присутствует постоянное мотивирование команды | Решения принимаются долго, недостаток дисциплины, походит только для маленьких проектов |
| «разработка, управляемая функциональностью» (fdd) | Пяти шаговый процесс, объектно-ориентированная разработка, очень короткие итерации (могут достигать по длительности несколько часов) | Простота метода, объектное моделирование | Данная методология фокусируется только на проектировании и внедрении, очень мало внимания уделено именно разработке |
| «канбан» | Самоорганизующаяся команда, утерянный смысл понятия «итерация» - весь упор на задачи | Нет ограничений по времени выполнения, зато есть ограничение на число «работы в данный момент», эффективно, когда неизвестно, что может в дальнейшем потребоваться заказчику от ПО | Недостаток дисциплины, затяжной характер |

FDD фокусируется на пяти шаговом подходе, который базируется на идентификации, разработке и внедрении характеристик. В FDD также полагается, что часть работы по проекту уже сделана. В результате, многие фазы проекта остаются не до конца реализованными. «Канбан» может быть эффективен внутри конкретной организации, даже внутри какого-то отдела, в случае сложных контрактных отношений «заказчик-исполнитель» он очень трудно реализуем. «Бережливая разработка» и «экстремальное программирование» тоже теряют свою эффективность, когда в проект вовлечено более чем одна организация: отсутствие явного наблюдателя за процессом может создать путаницу в обязанностях. Методология «скрам» нацелена на взаимодействие с заказчиком, и, несмотря на то, что команда разработчиков сама решает, какие задачи она будет выполнять в течение одной итерации, в данной методологии присутствует наблюдатель (Скрам-мастер), который контролирует соблюдение «скрам»-процесса. Помимо всего прочего, «скрам» хорошо документирован: существует специальное «руководство по скраму», переведенное на многие языки мира. Определенная степень формализованности в терминах «гибкой» методологии разработки является наилучшей стратегией управления IT-проектом, когда в проекте участвуют люди из разных организаций.

«Скрам» - наиболее популярная «гибкая» методология, широко используемая зарубежными компаниями, такими как Yahoo!, PayPal, Nike, Google, SAP, GE для управления проектами (Deemer, 2007). Согласно шестому и седьмому ежегодным государственным исследованиям «гибкой» разработки Интернет-ресурса Version One, методология «скрам» используется как метод «гибкого» управления IT-проектом в несколько раз чаще, чем остальные методологии (рисунок 8 и рисунок 9):[[6]](#footnote-6)[[7]](#footnote-7)

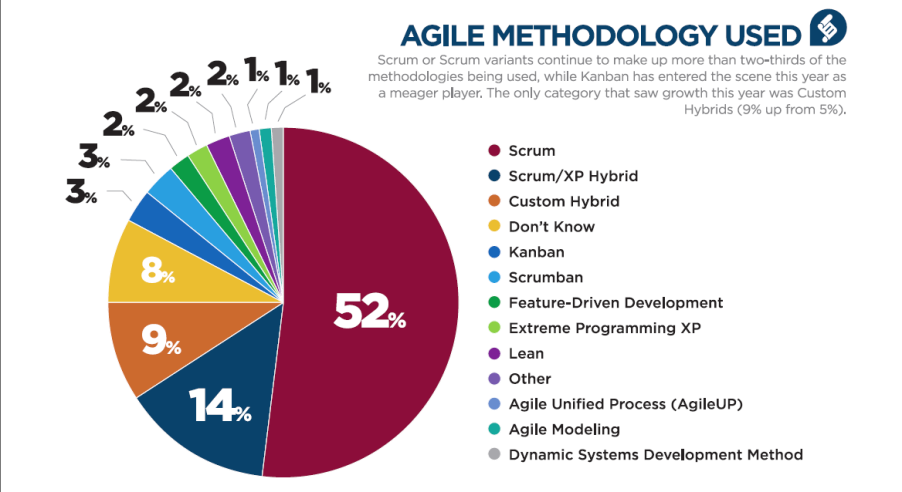


Рис. 8 Популярность методологии "скрам" в 2011 году

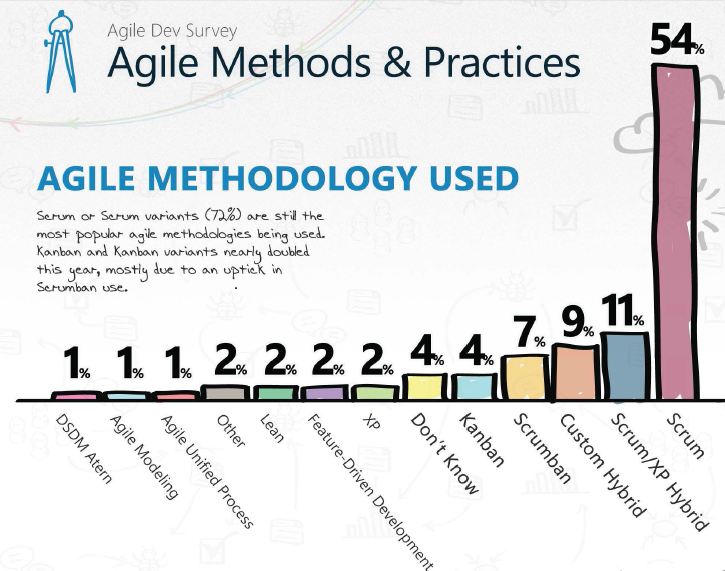


Рис. 9 Популярность методологии "скрам" в 2012 году

## 2.3. Описание методологии «скрам»

Методология «скрам» была впервые представлена в 1990-х годах Кеном Швабером и Джефом Сазерлендом в виде четко задокументированного и формализованного «руководства по «скраму».

«Скрам» - это гибкий и легкий процесс управления и контроля программным обеспечением и процесса разработки продукта в быстро меняющихся условиях» [28, с. 19]. Данная методология устанавливает определенные правила управления IT-проектом (разработкой или внедрением информационной технологии), которые основываются на возможности постоянной корректировки требований и на внесении тактических изменений.

Перед началом проекта участники обсуждают глобальные цели и базовую стратегию, направленную на достижение этих целей. Лицо со стороны Заказчика проекта представляет интересы своей компании путем описания бизнес-функций внедряемого или настраиваемого программного обеспечения. Со стороны исполнителя определяется состав команды разработчиков и другие ресурсы, после чего сторонами обсуждаются примерные рамки проекта, а так же дата начала первой итерации. Итерация в «скраме» получила название **спринт**,и она длится две-четыре недели. Все спринты имеют относительно фиксированную длительность: это означает, что если запланирована дата окончания спринта, то при наступлении этой даты (с минимальной погрешностью) все работы данного спринта должны закончиться, вне зависимости от того, успела ли команда проекта выполнить задание или нет. Минимальная погрешность означает, что на практике какие-то задержки все же возможны, но, тем не менее, они не должны превышать двух-трех дней для спринта, длительность которого равняется двум неделям.

Модель методологии «скрам» состоит из трех главных компонентов: ролей, артефактов и мероприятий. В данной методологии присутствуют три роли: Владелец продукта, Скрам-мастер и Скрам-команда.

* **Владелец продукта** чаще всего является заказчиком, который поддерживает в актуальном состоянии список требований к проекту. Чем лучше и четче владелец продукта опишет требования, тем меньше будет вопросов у разработчиков, тем реже будет изменяться функционал программного обеспечения с течением времени. Именно Владелец продукта определяет приоритет требований, то есть сам заказчик решает, какие функции программного обеспечения являются наиболее срочными и важными для компании на какой-то конкретный момент;
* Скрам-мастер ответственен за понимание сущности всего «скрам»-процесса другими участниками. Скрам-мастер чем-то похож на традиционного менеджера проекта, но у него есть одно отличие: он не отдает явных приказов и не устанавливает сроки разработчикам, он, напротив, помогает им при возникновении каких-либо трудностей и следит, чтобы их работа была слаженной. Именно от Скрам-мастера зависит атмосфера в команде разработчиков, а, как следствие, и их инициативность, удовлетворенность и общий результат. Скрам-мастер решает любые проблемы, которые могут как-то помешать команде, будь то сломанное оборудование или внешнее давление со стороны заказчика. Скрам-мастер также следит, чтобы все события «скрама» (о которых будет написано ниже) происходили регулярно и с максимальной эффективностью;
* Скрам-команда «…состоит из профессионалов, выполняющих работу по разработке потенциально «готового» к выпуску новой версии продукта в конце каждого «спринта» [36, с. 5]. Обычно количество разработчиков и программистов не превышает восьми, и все они являются заинтересованными в успешной реализации IT-проекта. Перед началом каждой итерации Скрам-команда ставит достижимую и значимую для заказчика цель, а во время итерации направляет все свои усилия на достижение этой цели в срок с заявленным качеством. Достижение цели характеризуется наличием запланированного кода, который впоследствии был отлажен, а возможные дефекты итерации устранены. Скрам-команда сама устанавливает себе сроки (обсудив их со Скрам-мастером и Владельцем продукта), сама оценивает свои возможности и распределяет время.

Среди артефактов методологии «скрам» выделяют журнал продукта, журнал спринта и диаграммы сгорания (рисунок 10):



Рис. 10 Артефакты методологии "скрам"

**Журнал продукта** создается Владельцем продукта (заказчиком) после анализа потребностей бизнеса в виде списка требований к программному обеспечению. Иными словами, заказчиком описываются главные желаемые функционалы разрабатываемой или внедряемой программы. После этого каждому требованию приписывается степень важности для заказчика, то есть приоритет, причем наиболее приоритетные задачи должны находиться выше по списку в журнале продукта, чем наименее приоритетные. В течение всего проекта Владелец продукта имеет право добавлять в журнал продукта новые требования или модифицировать старые. Именно такая возможность позволяет данному подходу быть «гибким к изменениям». В реальной жизни у заказчика может постоянно меняться список требований к программному обеспечению, соответственно, Владелец продукта может вести их непосредственный учет и служить «соединяющим» звеном между компанией-заказчиком, интересы которой Владелец продукта представляет, и Скрам-командой (разработчиками). Выбранные на определенный спринт требования из журнала продукта заносятся в **журнал спринта**.

Схематически это можно представить следующим образом (рисунок 11):

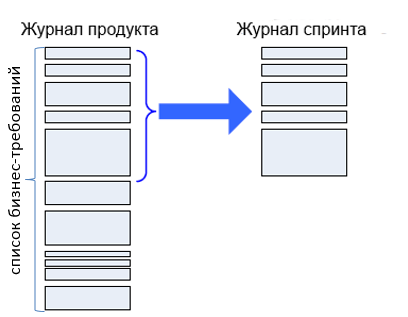


Рис. 11 Формирование журнала спринта

В отличие от журнала продукта, который может быть изменен или пополнен Владельцем продукта в любой момент времени проекта, журнал спринта на время конкретной итерации является фиксированным и утвержденным командой. Задачи из журнала спринта могут быть исключены только в форс-мажорных ситуациях, когда их выполнение больше не будет нацелено на успех всего IT-проекта. Как было сказано ранее, методология «скрам» вовсе не исключает документацию полностью, ее просто намного меньше, чем в традиционном подходе водопадной разработки. На практике все решается Владельцем продукта и Скрам-командой: если обе стороны приходят к соглашению оформить процесс реализации какой-то задачи из журнала продукта, то разработчиками создается соответствующий небольшой документ. Данный документ не будет являться подобием технического задания, на него не будут ссылаться при обсуждении контрактных условий, он необходим для более конкретного понимания функционирования каких-либо частей программы.

Для того чтобы вести учет объема выполненной работы, а также объема предстоящей работы, используются **диаграммы сгорания**. Перед каждой итерацией команда планирует количество усилий, которое необходимо для выполнения задач из журнала спринта. После завершения каждой задачи Скрам-мастер пересчитывает количество оставшейся работы на этот спринт. Эти значения отмечаются на графике, где по оси абсцисс находится длительность спринта, а по оси ординат количество оставшейся работы. Диаграмма сгорания полезна в качестве вспомогательного инструмента, позволяющая оценить, в правильном ли направлении идет работа команды в рамках конкретного спринта.

Помимо спринта в методологии «скрам» существуют еще четыре мероприятия: планирование спринта, ежедневное совещание, обзор итогов спринта и ретроспективное совещание. Перед началом спринта происходит его **планирование**. Сначала Владелец продукта демонстрирует журнал продукта, в котором находятся требования к программному обеспечению, отсортированные по приоритету на конкретный момент. При этом Владелец продукта обсуждает со Скрам-командой возможные нюансы задач, тем самым, минимизируя возможность появления субъективизма в разработке. После этого команда выбирает наиболее важные задачи из журнала продуктов, которые помещаются в журнал спринта. Ключевым момент здесь является определение необходимого времени для выполнения этих задач самими разработчиками. Так как в данном случае они все решают сами, без внешнего принуждения от менеджера проекта, то вероятность к неоправданному завышению времени минимальна. После того, как задачи на следующий спринт назначены, Скрам-команда начинает между собой обсуждать каждую из них более подробно. Для успешного выполнения задачи, она может детализироваться понятным разработчикам образом на мини-задачи. Кроме этого, Скрам-команда обсуждает сам процесс разработки кода, взаимозависимость между различными компонентами и так далее. Иными словами, разработчики решают, как они будут реализовывать требование заказчика. Методология «скрам» нацелена на организацию слаженной работы внутри команды. Планирование спринта длится около четырех часов и главным результатом этого мероприятия является договоренность между Владельцем продукта и Скрам-командой по поводу количества задач, над которыми будут вестись работы в течение планируемого спринта, а так же уверенность в том, что понимание пришло ко всем участникам встречи. Несмотря на «гибкость» методологии «скрам», команда должна быть уверена в неизменности и стабильности задач в рамках конкретного спринта. Если Владелец продукта хочет поменять требования, то для этого ему стоит дождаться планирования следующего спринта. Такой подход одновременно «гибок» и одновременно защищает проект от возможного беспорядка и разногласий. После окончания планирования начинается спринт. Каждый рабочий день Скрам-мастер организует короткое **совещание**, не превышающее по длительности пятнадцати минут. Целью **ежедневного совещания** является предоставление возможности участникам команды поделиться собственным прогрессом и возникшими трудностями. Каждый участник команды отвечает на три вопроса:

1. Что было сделано с момента прошлого такого совещания?
2. Что планируется сделать к следующему такому совещанию?
3. Какие существуют сложности, тормозящие процесс выполнения задания?

При возникновении каких-то проблем, Скрам-мастер делает все возможное, чтобы устранить их после окончания ежедневного совещания. Таким образом, пятнадцатиминутная встреча помогает держать в курсе участников Скрам-команды о том, что делают их коллеги и на какой стадии находятся задачи из журнала продукта. Владелец продукта тоже может присутствовать на этой встрече, однако никаких дискуссий между ним и разработчиками быть не должно: встреча организуется только для краткой отчетности (Вольфсон, 2012).

Если Скрам-команде не удалось достичь цели спринта, то на последнем ежедневном совещании разработчики признают этот факт и пытаются разобраться в причинах произошедшего. Вполне возможно, что на этапе планирования спринта неправильно были оценены возможности команды и требуемое время для выполнения задач. Очень часто это происходит на самых первых спринтах, что логично: команда еще не до конца могла сработаться с Владельцем продукта и с новым IT-проектом в целом. Однако в будущем такие неудачи должны быть исправлены с помощью нужного влияния Скрам-мастера.

После завершения спринта проводится **обзор его итогов**, где демонстрируется основная функциональность, появившаяся в проекте за этот спринт. На этой встрече может присутствовать любой желающий (например, кто-то из руководства компании-заказчика хочет увидеть новый функционал). Важно отметить, что обзор итогов спринта не предполагает под собой организацию и планирование презентации, на деле должна быть непосредственная демонстрация того, что было сделано в прошедшем спринте. Владелец продукта определяет полноту и качество выполненного задания, тем самым налаживая обратную связь с разработчиками (рисунок 12):

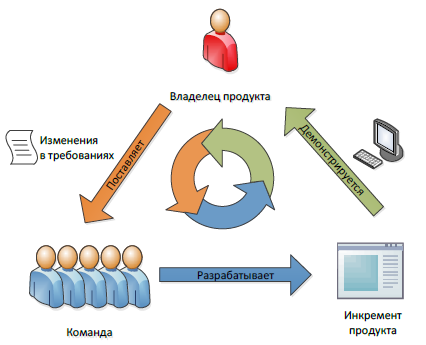


Рис. 12 Организация обратной связи в рамках обзора итогов спринта

Впоследствии Владельцем продукта корректируется журнал продукта и делаются некоторые выводы касательно дальнейших сроков проекта. Результатами встречи являются скорректированный журнал продукта и планы на дальнейший спринт.

**Ретроспективное совещание** также проводится по завершении спринта, однако на этой встречи присутствуют только участники Скрам-команды, Скрам-мастер и Владелец продукта (для дополнительной обратной связи).

Длительность совещания варьируется от пятнадцати минут до двух часов, в зависимости от того, долгий ли был спринт, много ли обнаружено проблем и так далее. Во время этой встречи разработчики обсуждают между собой проблемы и позитивные моменты прошедшего спринта. Если же команда приходит к выводу, что в проекте присутствует проблема, мешающая всем участникам, то выносится решение о изменениях, которые можно осуществить в следующем спринте для того чтобы улучшить рабочий процесс. Пример результатов ретроспективного совещания приведен на рисунке 13:



Рис. 13 Визуализация позитивных и негативных моментов во время ретроспективного совещания

В общем и целом, «скрам»-процесс может быть схематически представлен следующим образом (рисунок 14):



Рис. 14 Полноценный "скрам" процесс

Подводя итог, можно отметить, что благодаря **постоянному анализу выполненной работы** и **возможностям осуществлять корректировки направления проекта между итерациями (спринтами)** методология «скрам» позволяет более продуктивно достичь результатов и, следовательно, более **качественно** разработать программное обеспечение. Методология предполагает, что команда выбирает те задачи, которые являются наиболее приоритетными для бизнеса и соответствуют техническим возможностям. Кроме того, отсутствие внешнего давления и самостоятельность в выборе объема работы способствуют тому, что разработчики чувствуют себя полностью вовлеченными в процесс разработки специалистами, от которых зависит качество продукта, а не простыми исполнителями, которые лишь выполняют поручения своего менеджера. Такой подход стимулирует проявление энтузиазма и активности у разработчиков. Помимо всего прочего, самостоятельный выбор объема работ предполагает, что работник знает свою производительность, а значит вероятность отклонения от графика значительно ниже, чем при той же традиционной модели управления IT-проектом. От длительности зависит стоимость проекта, соответственно, если заказчику придется оплатить дополнительное время работы разработчика (если разработчик не уложился в свой собственно поставленный срок), то эта стоимость будет намного ниже стоимости запроса на изменение, который мог бы возникнуть в аналогичной ситуации в водопадной модели.

Методология «скрам» нацеливается на постоянное изменение приоритетов требований бизнеса, что увеличивает доходность проекта даже на самых ранних этапах. Кроме того, за счет постоянных встреч Скрам-команды и Владельца продукта, на которых участники получают обратную связь друг от друга, команде удается разрабатывать именно такое программное обеспечение, которое **максимально соответствует ожиданиям заказчика**. Так как в проекте важную роль играет удовлетворение заказчика, то IT-проект можно считать качественным не только когда он отлажено работает после внедрения, но и когда соответствует требованиям заказчика. Значительным преимуществом является возможность наблюдения за промежуточным продуктом, разработанным или внедренным в течение определенного спринта. Это позволяет **обнаруживать и исправлять ошибки внедренного фрагмента на ранних этапах**. В этом и заключается степень качества проекта. Владелец продукта вместе со своими коллегами может сам увидеть работу программы, пусть и с минимум возможностей.

# Глава III. Управление проектом внедрения специализированного программного обеспечения в Банк

## 3.1. Характеристика внедряемого программного обеспечения в Банк

Банком «XXX» внедрялось специализированное программное обеспечение для автоматизированной оценки кредитоспособности клиента и для принятия решения о выдаче кредита. В качестве заказчика разработки и внедрения ПО выступал непосредственно Банк, в качестве исполнителя – поставщик программного обеспечения для финансового рынка.

Внедряемая программа представляет собой промышленную платформу, предназначенную для проверки рисковых правил в процессе рассмотрения заявки на получение кредита. Внедряемая программа позволяет:

* Автоматизировать бизнес-процесс кредитования в Банке;
* Автоматизировать современные методы оценки риска клиента и заявки в соответствии с кредитной политикой Банка;
* Минимизировать время рассмотрения заявки на выдачу кредита;
* Уменьшить операционные риски Банка;
* Сформировать новое продуктовое предложение клиенту, соответствующее его риск-сегменту.

Внедрение такой программы в Банк сможет так же и сократить количество человеческого влияния, что, в результате, может минимизировать ошибки, связанные с погрешностью работников.

Для полноценной работы внедряемое программное обеспечение интегрируется с фронт-офисной системой, обеспечивающей корректную поддержку бизнес-процесса кредитования. В рамках рассмотрения одной заявки будет произведено несколько обращений к внедряемой системе с помощью пользовательского интерфейса. Обмен данными между системами осуществляется посредством обмена XML-сообщениями, в которые при каждом вызове дописывается новая информация. Схема XML-обмена приведена на рисунке 15:

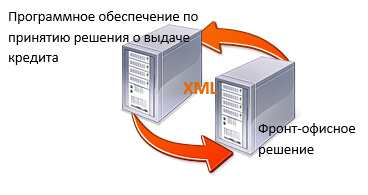


Рис. 15 Схема XML-обмена

После одного из последних обращений к системе XML-сообщение содержит в себе риск-сегмент заявки, после чего, согласно кредитной политике банка принимается решение о выдаче кредита.

## 3.2. Управление проектом внедрения ПО на основе водопадной модели

План проекта по внедрению автоматизированной системы по принятию решения о выдаче кредита составлялся на базе традиционной модели жизненного цикла проекта, то есть, в виде «водопада» - последовательных фаз, где каждая фаза не может быть начата до завершения предыдущей. Официальная дата начала проекта – 23 мая 2012 года.

Базовый план проекта и распределение ресурсов были построены в программе управления проектами **Microsoft Office Project** (рисунок 16):

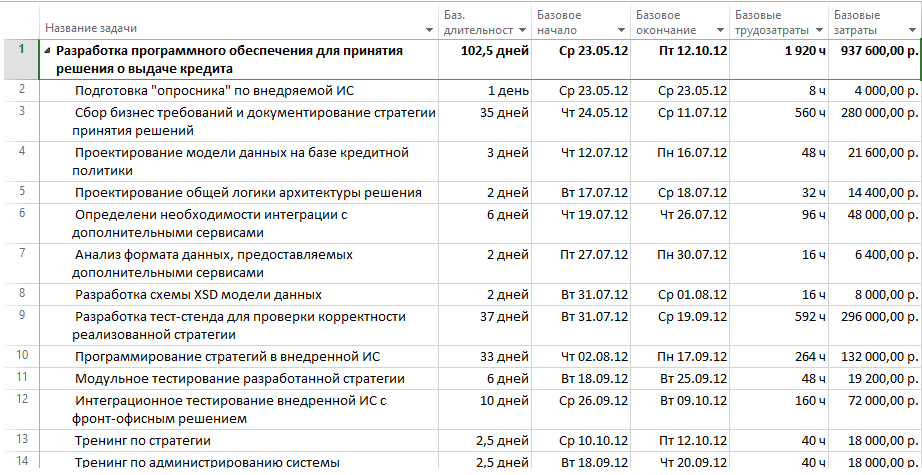


Рис. 16 Базовый план проекта по внедрению ПО в банк

Соответствующая базовому плану диаграмма Ганта (визуализированная цепочка задач) представлена на рисунке 17:

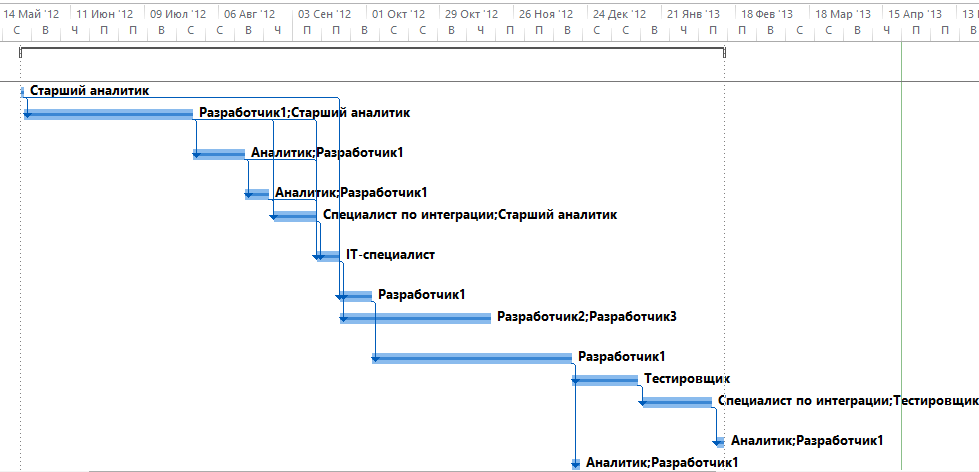


Рис. 17 Диаграмма Ганта проекта

Как видно из рисунка 16, проект был спланирован менеджером как классический «водопад», причем этапы сформулированы довольно широко. В представленном выше плане проекта отсутствуют операции слияния и практически отсутствуют дробящиеся операции.

После того, как менеджером проекта и командой проекта были начаты работы по внедрению программного обеспечения, оказалось, что сроки выполнения задач постоянно сдвигаются, причем вслед за задачей, выполняемой с увеличением базового срока, сдвигается начало выполнения всех последующих задач. Во время реализации проекта фиксировалась фактическая длительность каждой задачи, после чего был построен реальный календарный план.

Сравнение запланированных и фактических сроков выполнения проекта в процессе выполнения работ продемонстрировано на рисунке 18. Среди 14 задач 9 из них имели существенные отклонения по длительности: задача № 3, задача № 4, задача № 5, задача № 6, задача № 7, задача № 8, задача № 10, задача № 11, задача № 12. На рисунке они выделены цветом:

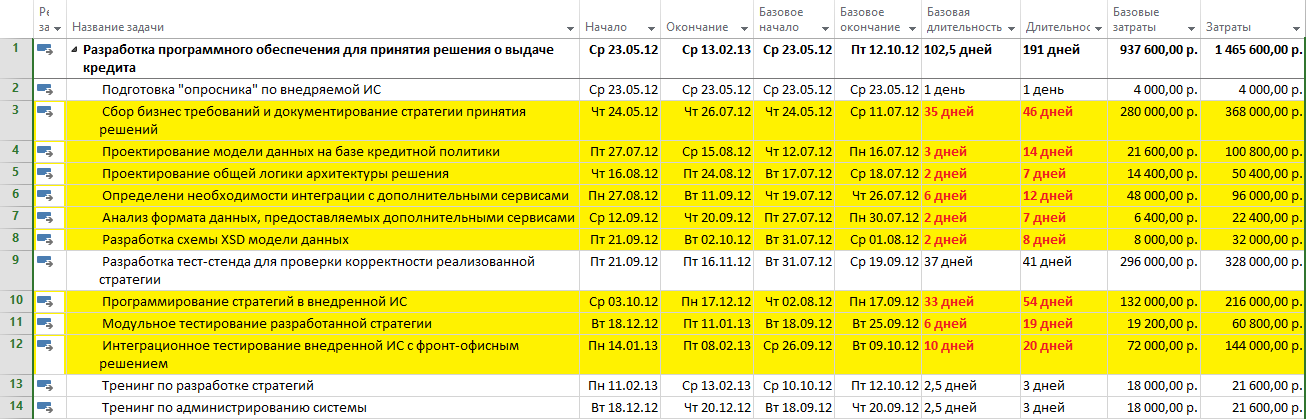


Рис. 18 Отклонения фактического плана от базового

Для наглядности отклонений была построена диаграмма Ганта с отслеживанием, где серым выделены задачи базового плана, а синим – задачи реального плана (рисунок 19). Даже по этому рисунку видно, что фактическая длительность проекта намного превышает запланированную. Этот же факт дает основания предполагать, что и стоимость проекта значительно увеличилась.

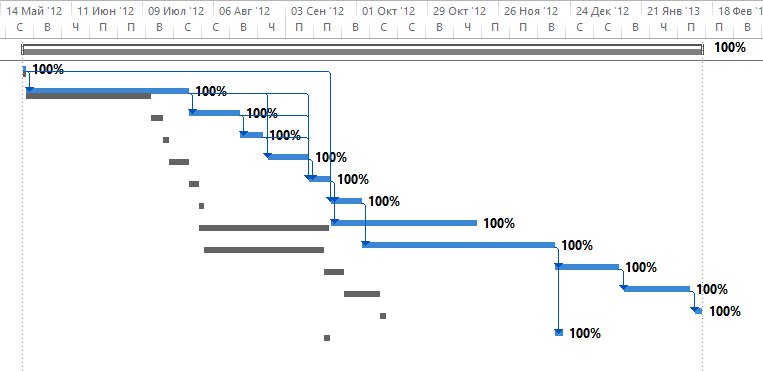


Рис. 19 Диаграмма Ганта с отслеживанием

С помощью наблюдения за ходом выполнения проекта были выявлены и систематизированы следующие причины задержек задач проекта:

**Задача № 3 «Сбор бизнес требований и документирование стратегии принятия решений»**.

Базовая длительность предполагалась равной 35 дням, фактическая составила 46 дней.

Данная задача является одним из основных этапов проекта, на котором оформлялось техническое задание на бизнес-спецификацию правил настройки рисковой стратегии. Со стороны исполнителя на данном этапе участвовали старший аналитик и основной разработчик.

* Долгое формирование требований, стремление к минимизации рисков путем полной аналитики возможных разногласий в будущем. Как следствие – задержка момента подписания технического задания со стороны Банка (заказчика);
* Вносились корректировки и добавления в рисковую политику, а как следствие, в требования к программному обеспечению;
* Долгая неопределенность операционного отдела в предоставлении своих требований аналитикам по работе внедряемого программного обеспечения;
* За задержки с Банка не взималась дополнительная плата, так как на тот момент не было подписано техническое задание (оно стало результатом данного этапа).

**Задача №4 «Проектирование модели данных на базе кредитной политики»:**

Базовая длительность предполагалась равной 3 дням, фактическая составила 14 дней.

* Во время предыдущего этапа отдел маркетинга не участвовал в формировании требований к программному обеспечению, но на текущем этапе его представители решили внести собственные пожелания по работе внедряемой программы с клиентом. Эти изменения в требованиях, по словам исполнителя, в некоторых местах противоречили ТЗ, поэтому за доработку этих требований Банк заплатил 79 200 рублей.

**Задача №6 «Проектирование общей логики архитектуры решения»:**

Базовая длительность предполагалась равной 6 дням, фактическая составила 12 дней.

* Задержки на этом этапе связаны с коммуникацией между исполнителем и компаниями, которые предоставляли дополнительные сервисы (CreditRegistry, ЦФТ).

**Задача №8 «Разработка схемы XSD модели данных»:**

Базовая длительность предполагалась равной 2 дням, фактическая составила 8 дней.

* На данном этапе разработчику пришлось изменить первоначальную модель данных, так как операционный отдел и отдел маркетинга Банка запросили новый функционал, необходимый им для работы с клиентом (возможность отсылать заявку клиента на предыдущий этап). Запрос на изменение стоил банку 24 000 рублей.

**Задача №9 «Разработка тест-стенда для проверки корректности реализованной стратегии»:**

Базовая длительность предполагалась равной 37 дням, фактическая составила 41 дней.

* Тест-стенд получает на вход XML-сообщение, затем обращается к серверу, где находится программное обеспечение по принятию решения о выдаче кредита, обрабатывающее запрос, и после этого на экран возвращается новое, дополненное XML-сообщение с результатами проверки. Тест-стенд в своем первом релизе не имел функционала сохранения XML-ответов, что показалось крайне неудобным Банку. Проблема задержки на данном этапе вызвана разным толкованием ТЗ разработчиками и работниками банка. Исполнитель считал, что задача выполнена корректно, по крайней мере предмет спора не был описан в ТЗ, в то время как Банк посчитал что такой очевидный пункт не стоило даже подробно описывать. Дополнительные работы исполнитель оценил в 32 000 рублей.

**Задача № 10 «Программирование стратегий во внедренной ИС»:**

Базовая длительность предполагалась равной 33 дням, фактическая составила 54 дней.

* Задержки на данном этапе связаны с явными изменениями бизнес требований программного обеспечения, вызванными корректировками отдела маркетинга, операционного отдела и отдела рисков. Многие из нововведений повлекли за собой изменения в модели данных (андеррайтеры из операционного отдела захотели видеть на интерфейсе распределение дохода по месяцам, произошло уточнение функционала некоторых кнопок на экранных формах, например, появилась возможность осуществления визуальной оценки клиента). Запросы на изменения на текущем этапе обошлись Банку в 84 000 рублей.

**Задача № 11 «Модульное тестирование разработанной стратегии»:**

Базовая длительность предполагалась равной 6 дням, фактическая вышла 19 дней.

* С результатами тестирования были ознакомлены сотрудники отдела рисков Банка. Задержки во время выполнения данной задачи вызваны расхождением между разработчиками программы и сотрудниками отдела рисков по поводу корректности настроенной стратегии. Заказчиком были установлены новые требования, учитывающие динамику заявки клиента во времени (в ТЗ данный функционал описан не был). Трудозатраты на добавление такого функционала были оценены менеджером проекта в 32 часа, что в денежном выражении равно 16 000 рублям согласно ставке разработчика.

**Задача №12: «Интеграционное тестирование внедренной ИС с фронт-офисным решением»:**

Базовая длительность предполагалась равной 10 дням, фактическая составила 20 дней.

* Так как на данном этапе в качестве человеческого ресурса используется специалист по интеграции, то очевидно, что он в своей работе руководствуется только техническим заданием, созданным задолго до этого этапа аналитиками и подписанным банком. Субъективная трактовка некоторых пунктов ТЗ привела к интеграционной реализации, которая не удовлетворяла Банк. В основном, разногласия были связаны с качеством и количеством отображаемых данных на экранной форме из внедряемой автоматизированной системы принятия решения (подсчет рисковых показателей, отображение доходов и обязательств клиента, результаты проверки его кредитных историй, результаты пробивки по черным спискам, отображающийся скоринговый балл);
* Внедряемое программное обеспечение рассчитывает рисковые показатели, риск-сегмент клиента, а также проверяет соответствие правилам кредитной политики банка. Вышеописанные действия предполагают формирование некоторых комментариев, которые в бизнес требованиях и ТЗ записаны на русском языке (пример «Клиент имеет негативную кредитную историю»). Их реализация была осуществлена на этапе программирования стратегии, однако на этапе интеграции с фронт-офисным решением выяснилось, что передача данных возможна только с помощью латинского алфавита; таким образом, разработчикам пришлось вводить понятие «транслитерации» для решения данной проблемы. Существенное время было потрачено на обсуждение проблемы между заказчиком проекта (Банком) и менеджером проекта.
* Издержки, понесенные Банком на данном этапе, составили 72 000 рублей.

Проанализировав целиком проект внедрения программного обеспечения в банк, можно отметить следующее:

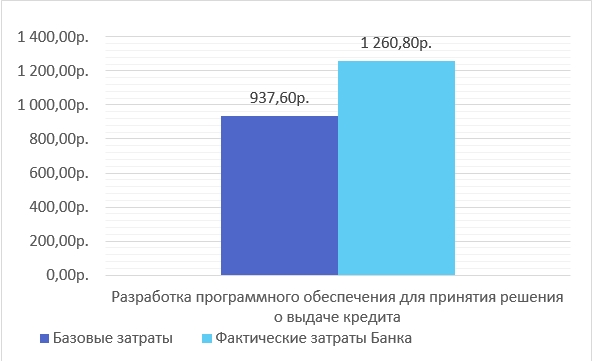
1. Главным критерием успешности проекта являлось качество. **Под качеством будет подразумеваться возможность выявления и исправления ошибок на ранних этапах проекта, а также полное соответствие результатов проекта требованиям заказчика.** Так как внедряемое программное обеспечение в будущем должно автоматически оценивать клиентов банка, при внедрении проекта было необходимо минимизировать операционные риски, которые могли бы возникнуть в дальнейшем. Как результат, длительность и бюджет проекта автоматически увеличивались из-за постоянных доработок программного обеспечения. Это означает, что для заказчика качество было ключевым моментом, ради которого он готов был пожертвовать длительностью и бюджетом;
2. В середине и конце проекта команда разработчиков находилась в постоянно стрессовом состоянии, так как отсутствие коммуникации с другими работниками повлекло за собой субъективное понимание технического задания. В результате менеджером проекта было решено назначить им дополнительный объем работу за счет Банка, но, несмотря на это, производительность работников была ниже предполагаемой из-за человеческого фактора;
3. Отклонение по времени составляет 88,5 дней (в базовом плане предполагалось 102,5 дней, фактически вышло 191 день);
4. В плане предполагалась стоимость проекта равная 937 600 рублям, фактическая составила 1 465 600 рублей, из которых 323 200 рублей – запросы на изменения, соответственно, **стоимость проекта для Банка равняется 1 260 800 рублям (рисунок 20)**:  
    

Рис. 20 Сравнение затрат

Ввиду специфики водопадной модели, на основании которой проектировался жизненный цикл этого IT-проекта, тестирование и отладка программного обеспечения происходит намного позже разработки, что автоматически исключает возможность обнаружения ошибок на ранних этапах и их дальнейшее исправление. Поэтому критерий качества в данном случае зависит напрямую от удовлетворения заказчика от полученного, в конечном счете, продукта.

Таким образом, чтобы действительно достичь более или менее желаемый уровень **качества**, длительность проекта пришлось увеличить на 86,3%, а стоимость на 34,5%.

Введем следующие обозначения:

,

.

Банк и менеджер проекта внедрения (представитель компании-Исполнителя) стремятся минимизировать возможные, **нежелательные отклонения** от **требуемого качества (то есть, целевой результат подразумевает соответствие продукта ожиданиям заказчика)**, следовательно, для того, чтобы минимизировать отклонения по качеству (которое было выбрано наиболее приоритетным из всех критериев проекта), длительность проекта и его бюджет пришлось увеличить:

## 3.3. Моделирование и анализ внедрения проекта с помощью «гибкой» методологии «скрам»

Проанализировав недостатки водопадной модели при ее использовании для управления IT-проектами в банке, а также преимущества «гибких» методологий разработки и выбрав методологию «скрам» как наиболее подходящую методологию для управления IT-проектом ввиду ее формализации и организованности, была смоделирована ситуация внедрения программного обеспечения для принятия решения о выдаче кредита с использованием методологии «скрам».

**Процесс моделирования можно описать следующим образом:**

(1) Вводятся предпосылки, уравновешивающие проектируемую модель с реальной ситуацией:

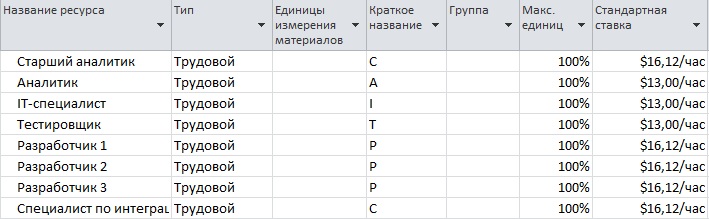
1. Производительности команд при использовании методологии «скрам» и водопадной модели равны;
2. Количество специалистов той или иной области в двух проектах одинаково, и оно приведено на рисунке 21:  
     
   

Рис. 21 Ресурсы

1. Время выполнения задач при использовании методологии «скрам» включает время проведения командой совещаний, что является тоже оплачиваемым заказчиком;
2. Сбор новых требований не занимает дополнительного времени, а происходит в течение «спринта»;
3. Внедрение программы происходит непосредственно в офисе банка, то есть для качественного применения методологии «скрам» лучше избегать удаленной работы.

(2) Описывается хронология ведения проекта и формализация действий:

Перед официальным началом проекта организовывается встреча представителя со стороны заказчика (Владельца продукта), которым является сотрудник отдела рисков, Скрам-мастера и Скрам-команды для обсуждения общего видения проекта. Обсуждается программа, которая будет внедряться, Скрам-командой происходит осознание базовых моментов, которые желает наблюдать заказчик. Заказчик, в свою очередь, пытается понять специфику программного обеспечения и принципы работы Скрам-команды. В течение этого времени (7-ми дней) происходит обмен общими договоренностями и обещаниями (например, участники проекта приходят к выводу о документировании каких-либо справочных значений).

«Скрам»-проект будет проходить следующим образом:

* Владелец Продукта подготавливает к каждому спринту журнал продукта, отсортированный по приоритету;
* Каждый спринт начинается с его планирования, т.е. команда определяет количество времени, которое необходимо каждому участнику для выполнения заданий из журнала продукта;
* Выбранные задания из журнала продукта переносятся в журнал спринта и, если надо, детализируются разработчиками;
* Во время планирования спринта участники команды обсуждают общую идею выполнения задач, их коррелированность с другими специалистами и прочее;
* В течение спринта команда выполняет задания, каждый день отчитываясь друг другу о выполненной работе во время ежедневного совещания;
* Выполнение сроков совещаний, а также поддержание атмосферы в команде обеспечивает Скрам-мастер;
* В конце спринта команда демонстрирует настроенный функционал Скрам-мастеру, Владельцу продукта и любым желающим со стороны заказчика (например, топ-менеджменту Банка);
* Если Скрам-команда не успевает сделать задание за один спринт, то, в случае мелких недочетов, спринт может задержаться на пару дней, а в случае крупных недоработок это же задание выносится снова в журнал продукта;
* В течение спринта у Владельца продукта могут появиться новые требования к программному обеспечению, которые он должен записывать в журнал продукта, после чего он дожидается встречи по планированию нового спринта для того, чтобы внести изменения в работу.

(3) Оценка базовой длительности задач, описанных с помощью методологии «скрам»:

Согласно методологии «скрам» команда разработчиков сама оценивает время, которое им может потребоваться для выполнения задач. Для моделирования проекта был проведен экспертный опрос разработчиков рассматриваемого IT-проекта для выявления предполагаемой длительности и максимально возможных отклонений. Им была объяснена идея внедрения программного обеспечения по итерациям с помощью методологии «скрам» и был продемонстрирован пример журнала продукта с базовыми требованиями от заказчика, отсортированный по убыванию важности для бизнеса на момент составления:

1. Способность внедряемой системы проверять клиента по минимальным требованиям кредитной политики;
2. Настройка функционала, предназначенного для тестирования настроенной стратегии во внедряемом программном обеспечении;
3. Способность внедряемой системы проверять и оценивать внешнюю кредитную историю клиента;
4. Способность внедряемой системы осуществлять поиск клиента по «черным» спискам (список неплательщиков и клиентов, являющихся должниками);
5. Способность внедряемой системы считать скоринговый балл по формуле, определенной Владельцем продукта;
6. Способность внедряемой системы анализировать результаты телефонной верификации клиента;
7. Способность внедряемой системы определять риск-сегмент заявки;
8. Способность внедряемой системы анализировать результаты проверки клиента по службе безопасности;
9. Способность внедряемой системы определять необходимость пересмотра заявки при изменении клиентом одобренных условий;
10. Способность внедряемой системы определять роль для окончательного принятия решения;

Иными словами Владелец продукта представил ряд функций, которые должно выполнять программное обеспечение. Как правило, это функции проверки данных клиента, взятые из анкетной информации и из внешних источников. Для Владельца продукта приоритетнее всего сначала настроить базовые проверки, а после этого осуществить настройку агрегированных (например, определение риск-сегмента заявки на основе полученных данных и проанализированных результатов).

При проведении опроса разработчики были предупреждены, что каждая итерация длится 14-30 дней, на что они впоследствии делали упор при моделировании. Таким образом, на самую первую итерацию (то есть, в журнал спринта) разработчики назначили первую задачу из журнала продукта и часть второй (рисунок 22). План проекта строился с помощью дополнения к программному обеспечению **Microsoft Office Project 2010**, которое называется **Scrum Tool,** которое просто помогает формализовать процесс.

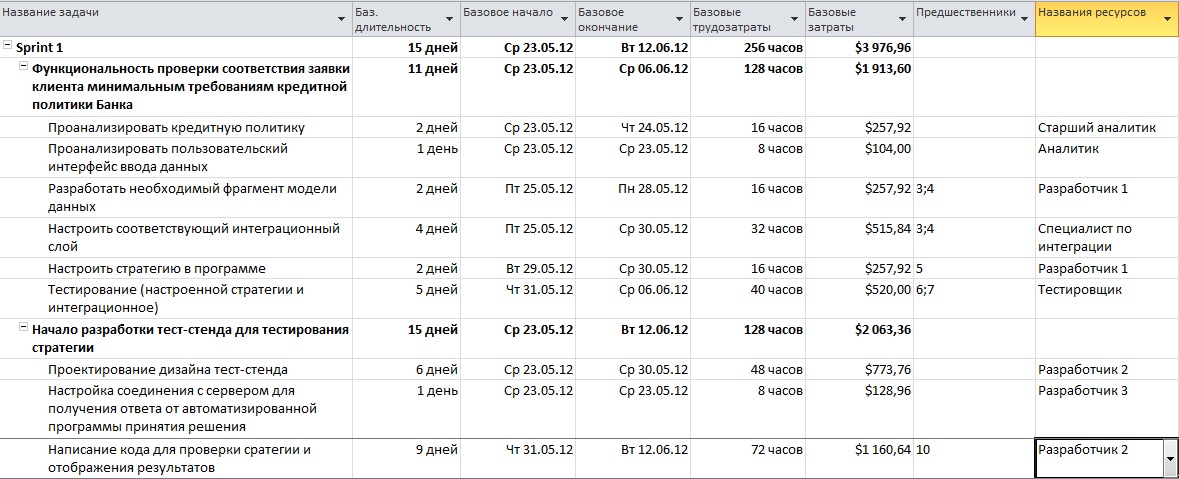


Рис. 22 Моделирование первого спринта

Как видно из выше представленного рисунка, в течение первого спринта были задействованы три разработчика, два аналитика и один специалист по интеграции. Вместе они детализировали требования из журнала спринта и оценили время, которое им требуется для выполнения своей части задания.

С учетом важных изменений в требованиях, которые были описаны при рассмотрении плана проекта, построенного с помощью водопадной модели, был смоделирован финальный журнал продукта (после завершения проекта) (рисунок 23):

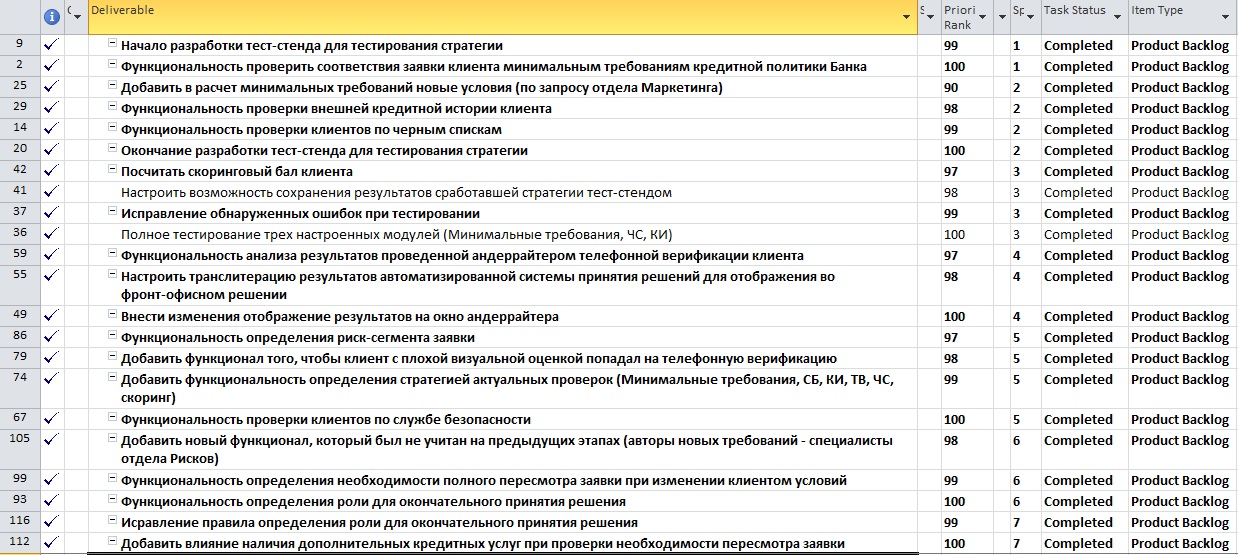


Рис. 23 Состояние журнала продукта в конце проекта

Как видно из рисунков, помимо названия требования-задачи в журнал пишутся так же степень приоритета, распределение в спринты (требования из журнала продукта отсортированы по хронологии проекта, т.е. по мере возникновения новых итераций-спринтов), и статус требования-задачи.

Важно отметить, что журнал продукта представляет собой требования заказчика, написанные понятным ему языком, («функциональность проверки внешней кредитной истории клиента», «добавить влияние наличия дополнительных кредитных услуг при проверки необходимости пересмотра заявки»), в отличие от этапов водопадной модели, составленных менеджером проекта, который больше ориентировался на понимание этапов своей командой («разработка схемы XSD модели данных», «программирование стратегии в ИС»). Подробное описание бизнес-требований снижает риск отклонения от желаемого заказчиком качества, когда речь идет о соответствии его требованиям (т. е. ).

Таким образом, смоделированный полноценный проект приведен на рисунке 24 (проект с детализацией задач приведен в Приложении 1):

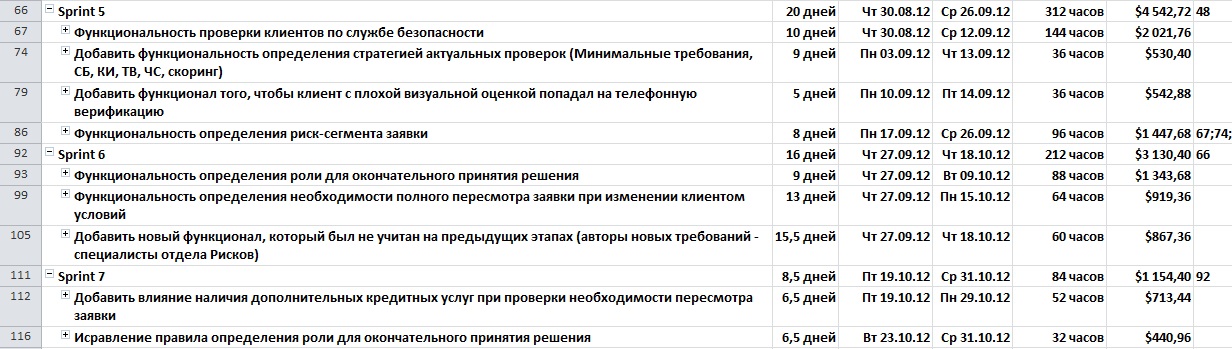
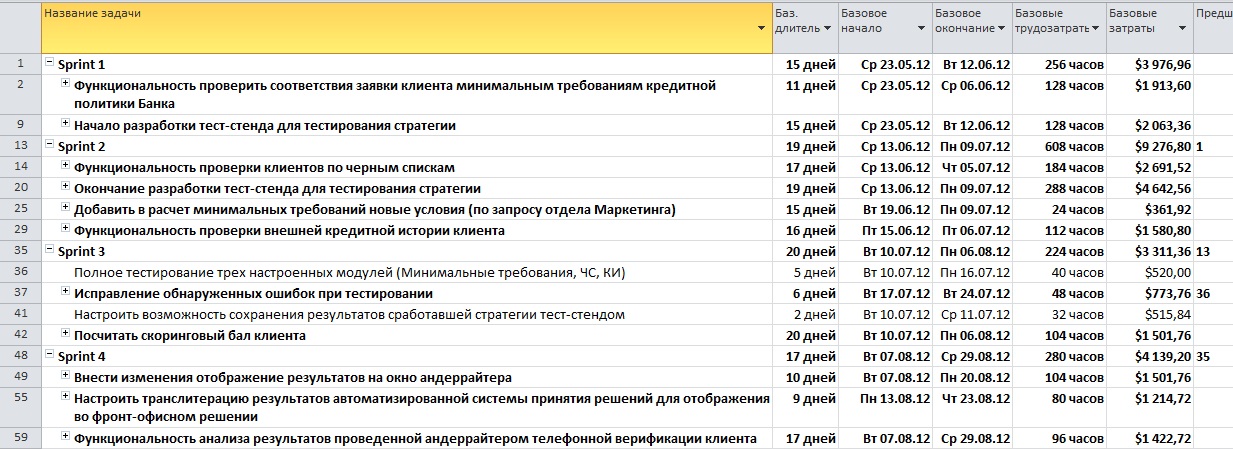


Рис. 24 Представление проекта внедрения ПО в банк с помощью «скрам»-методологии

Как видно из примера, все итерации очень похожи между собой, что было описано во второй главе работы: каждая итерация – это, своего рода, мини-проект настройки определенного функционала.

(4) Оценка максимально возможных отклонений по длительности задач, описанных с помощью методологии «скрам»:

На основании экспертного мнения разработчиков были смоделированы максимально возможные отклонения, доказывающие, что, в любом случае, добиться идеального проекта невозможно, но можно снизить вероятность задержек путем проведения постоянного анализа и выпуска инкрементов продукта (что и предполагает методология «скрам»). Помимо всего прочего, при использовании «скрам»-методологии возможно организовывать работу параллельно, что не позволяет сделать классический водопадный подход. Конкретный IT-проект, жизненный цикл которого был представлен с помощью водопадной модели, не имеет детализации задач, которые сами по себе описаны довольно обширно. На основании этих факторов разработчики, опираясь на свой экспертный опыт, предложили возможные максимальные отклонения, большая часть которых приходится на этапы, посвященные тестированию.

Смоделированные отклонения фактического плана от базового, построенного по методологии «скрам» представлены на рисунке 25:

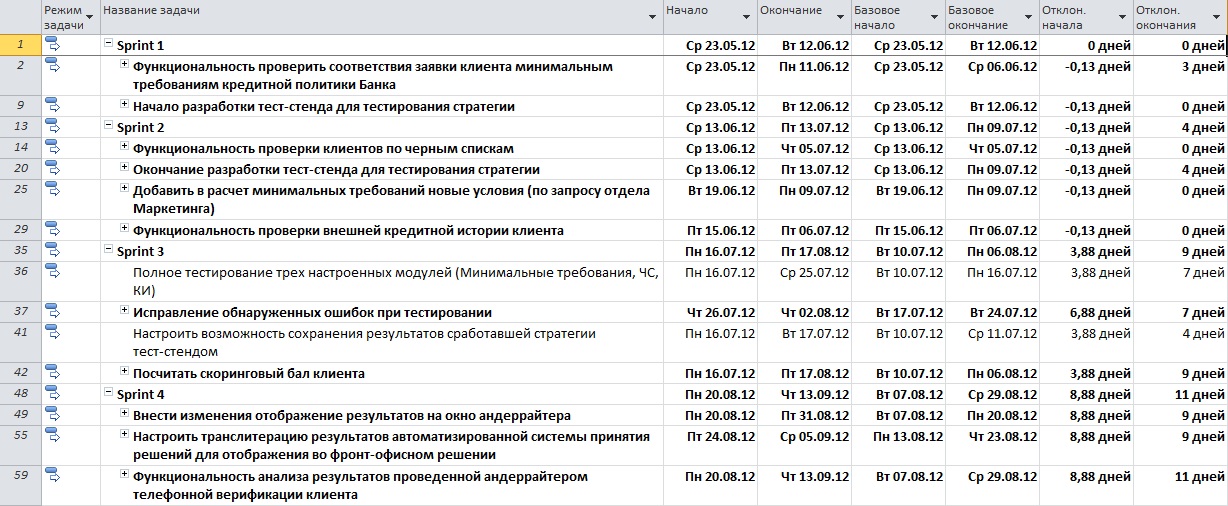


Рис. 25 Смоделированные отклонения фактического плана от базового по методологии "скрам"

(5) Сравнение базового и фактического планов:

Проанализировав длительности базового и фактических планов, можно заметить, что отклонение по длительности 18,6% (.

Так как проект, смоделированный с помощью методологии «скрам», не предполагает наличие запросов на изменение, время выполнения которых оценивается в водопадной модели менеджером проекта, важно отметить, что отклонения по стоимости проекта с предлагаемой методологией невелики, так как происходит только доплата за дни задержек. Базовая стоимость проекта оценивается в 915 487 рублей, а фактическая – в 1 076 171 рубль, таким образом, отклонение по стоимости (рисунок 26):

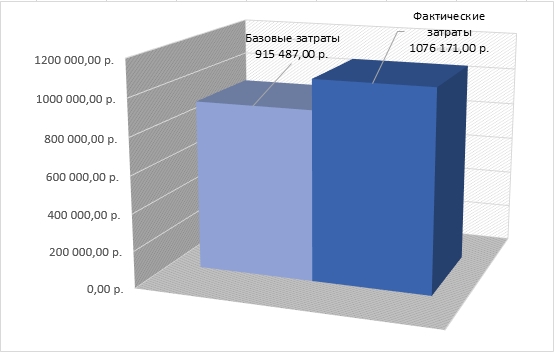


Рис. 26 Сравнение затрат

Как было неоднократно отмечено ранее, итеративный метод управления проектами хорош тем, что уточнение и демонстрация очередной версии ПО происходит довольно часто, после окончания очередной итерации. При таком подходе, очевидно, что возникающие на ранних этапах ошибки могут быть сразу же исправлены (в отличие от водопадной модели, где ошибки можно обнаружить только на этапе тестирование). Кроме того, получаемый в конце проекта продукт больше соответствует требованиям заказчика, чем «черный ящик», как в случае водопада. Соответственно, оперируя данными суждениями, можно отметить, что отклонение по качеству продукта, полученного с помощью методологии «скрам» намного меньше отклонения по качеству продукта, полученного с помощью водопадной модели:

В таблице 3 представлено соотношение общих критериев двух моделей: водопадной и «скрам»:

Таблица 3 Сравнение показателей

| **Модель** | **Качество** | **Длительность (дни)** | | **Отклонение (%)** | **Стоимость (руб)** | | **Отклонение (%)** |
| --- | --- | --- | --- | --- | --- | --- | --- |
| **Базовая** | **Фактическая** | **Сметная** | **Фактическая** |
| Водопадная |  | 102,5 | 191 | 86,3% | 937 600 | 1 260 800 | 34,5% |
| «Скрам» |  | 123 | 146 | 18,6% | 915 487 | 1 076 171 | 17,5% |

Несмотря на то, что базовая длительность проекта, смоделированного с помощью методологии «скрам» больше базовой длительности проекта, реализованного с помощью водопадной модели, отклонения и по длительности и по стоимости у проекта, смоделированного с помощью методологии «скрам» намного меньше, чем у водопадной модели при очевидном выигрыше в качестве внедренного ПО.

# Заключение

Таким образом, после проведения исследования можно выделить следующие результаты:

1. Были проанализированы определения терминов «проект» и «IT-проект» для выявления отличительных характеристик последнего. Было выяснено, что управление IT-проектом намного отличается от управления проектом, не связанным с информационными технологиями.
2. Были проанализированы критерии-показатели, свидетельствующие об эффективности проекта (бюджет, длительность, область охвата, качество); было выявлено, что изменение одного из этих показателей имеет существенное влияние на все остальные;
3. Было рассмотрено понятие «жизненный цикл IT-проекта» и были проанализированы преимущества и недостатки основных моделей: водопадной (традиционной), итеративной и спиральной. В результате было выяснено, что сильные стороны итеративной и спиральной моделей стали основой создания «гибких» методологий;
4. Была проанализирована специфика предметной области исследования (банковской сферы) на предмет применимости той или иной модели жизненного цикла при управлении IT-проектом. Как следствие, была выдвинута гипотеза о наиболее эффективном применении «гибкой» методологии управления IT-проектом, которая является улучшенным синтезом итеративной и спиральной моделей;
5. В результате рассмотрения характеристик и преимуществ «гибких» методологий управления IT-проектами, а также после сравнительного анализа видов «гибких» методологий, была выбрана методология «скрам», являющаяся наиболее четко прописанной и более подходящей для отношений вида «заказчик-исполнитель»;
6. В работе был разработан базовый план проекта внедрения специализированного программного обеспечения для автоматизированной оценки кредитоспособности клиента и для принятия решения о выдаче кредита в Банк «XXX», основанный на водопадной модели жизненного цикла. После сравнения сроков и стоимости базового проекта с фактически реализованным было выявлено, что для минимизации возможных отклонений по требуемому заказчиком качеству ПО длительность проекта была увеличена на 86,3% а стоимость на 34,5%.
7. Для моделирования проекта по предложенной методологии «скрам» был проведен опрос у команды разработчиков, с целью определения ими длительности выполнения задач и максимально возможных отклонений. По этой методологии «скрам» был смоделирован проект внедрения специализированного программного обеспечения в Банк. Также был смоделирован план «скрам»-проекта, удовлетворяющий качеству и требованиям заказчика с учетом максимально возможных отклонений по длительности решения задач. В результате отклонения по длительности выполнения проекта составили 18,6%, а по стоимости 17,5% при практически минимальных отклонениях от качества.

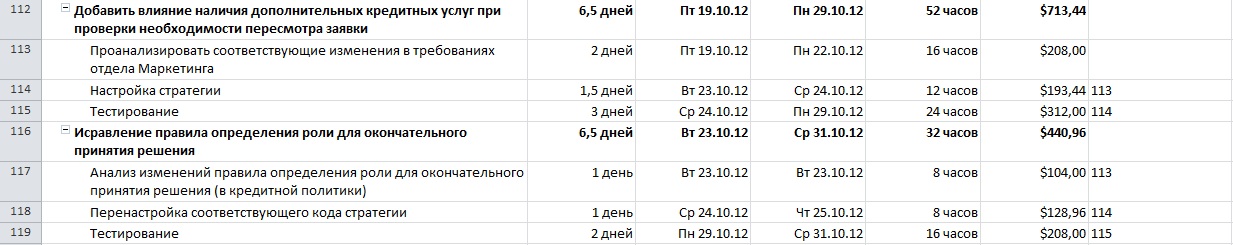
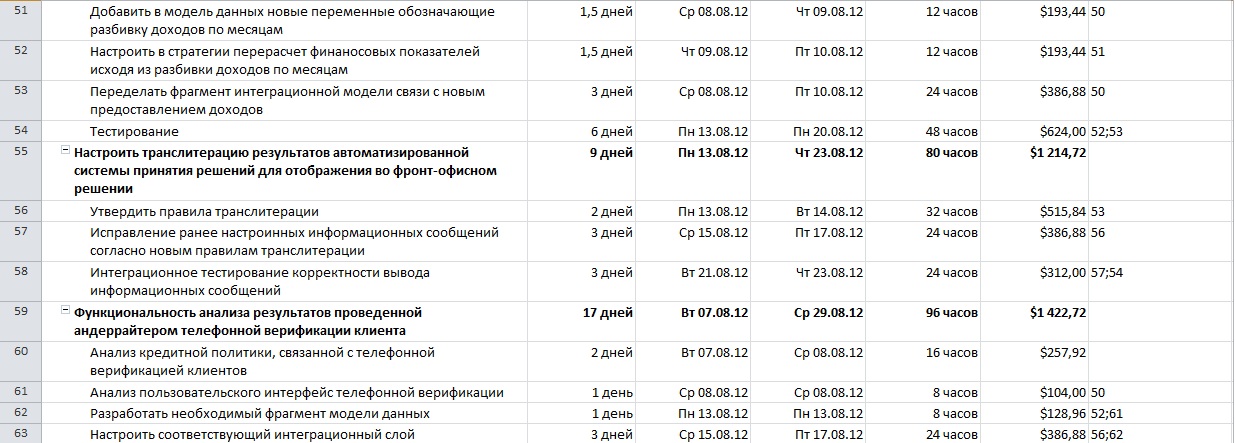
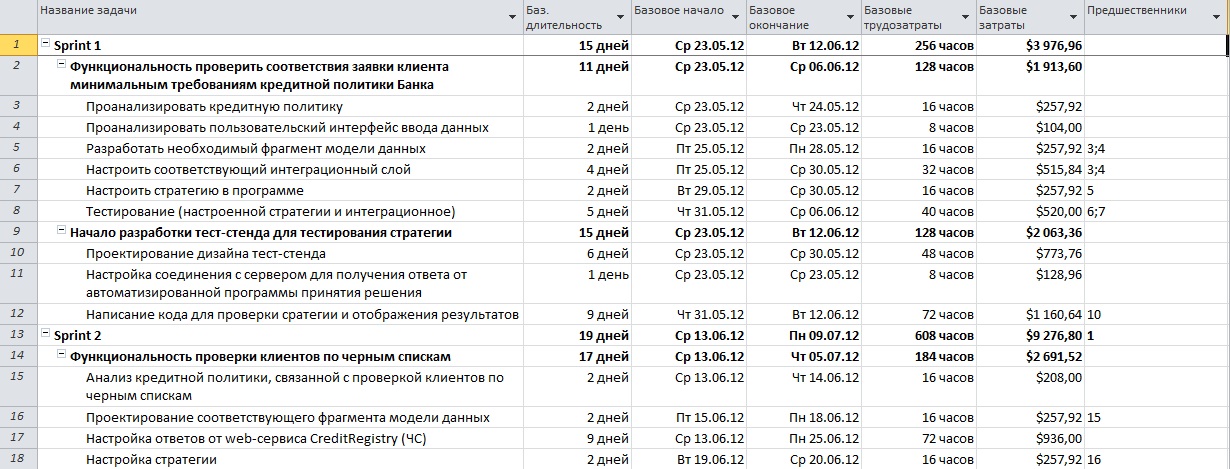
Проведенный анализ показывает, что в условиях быстро меняющихся и нечетко определенных требованиях (такие условия как раз-таки присуще банковской сфере), а так же при низком уровне коммуникации между отделами компании, использование методологии «скрам» в управлении IT-проектами является наиболее эффективным как для вендора, так и для заказчика. Говоря о вендоре, стоит отметить, что слаженная работа, которая определяется самими специалистами, выполняющими ее, намного больше стимулирует исполнителей на качественное решение задач, нежели постоянные поручения менеджера проекта и пребывание в вечном стрессе. Для заказчика выгода очевидна: проект внедряется быстрее, качественнее и с меньшими затратами по сравнению с традиционным методом.

# Список использованной литературы

1. Богданов В. Управление проектами. Корпоративная система – шаг за шагом. – М.: Манн, Иванов и Фербер, 2012. – 248 с.
2. Брукс Фредерик. Мифический человеко-месяц, или как создаются программные системы. – М, Символ-плюс, 2010. – 304 с.
3. Вольфсон Б. Гибкие методологии разработки, версия 1.2 (электронная книга), 2012.
4. Грей, Клиффорд Ф. Управление проектами: учебник: пер. с англ. третьего, полн. перераб. изд./ Клиффорд Ф. Грей, Эрик У. Ларсон; [науч. Ред. перевода В. М. Дудников]. – М.: Издательство «Дело и Сервис», 2007. – 608 с. – Доп. тит. л. англ.
5. Грекул В. И., Коровкина Н. Л., Куприянов Ю. В. Методические основы управления ИТ-проектами. - Интернет-университет информационных технологий, 2011. – 392 с.
6. Гробовцов Г. Я. УПРАВЛЕНИЕ ПРОЕКТОМ: Учебно-методический комплекс. – М.: Изд. центр ЕАОИ, 2009. – 288 c.
7. Дитхелм Г. Управление проектами. В 2 т. Т. I. пер. с нем. – СПБ.: Издательский дом «Бизнес-пресса», 2004. – 400 с.
8. Дитхелм Г. Управление проектами. В 2 т. Т. II. пер. с нем. – СПБ.: Издательский дом «Бизнес-пресса», 2004. – 288 с.
9. Мазур И. И., Шапиро В. Д., Ольдерогге Н. Г. Управление проектами: Учебное пособие / Под общ. ред. И.И. Мазура. — 2-е изд. — М.: Омега-Л, 2004. — с. 664
10. Портни, Стэнли И. Управление проектами для «чайников».: Пер. с англ. – М.: Издательский дом «Вилямс», 2005. – 352 с.: ил. – Парал. тит. англ.
11. Разу М. А., Бронникова Т. М., Разу Б. М., Титов С. А., Якутин Ю. В. Управление проектом. Основы проектного управления: учебник / кол. авт,: под ред. проф. М. А. Разу. – М.: КНОРУС, 2006 – 768 с.
12. Товб А. С., Ципес Г. Л. Управление проектами: стандарты, методы, опыт. – М.: ЗАО «Олимп-Бизнес», 2003. – 240 с.: ил.
13. Грекул В. Проектирование информационных систем (лекции), 2010
14. Черных Е. А. Agile Project Management и его история, М., «Менеджмент качества», 02(02)2008
15. Pekka Abrahamsson, Outi Salo, Jussi Ronkainen and Juhani Warsta. Agile Software Development Methods - Review and Analysis, by Technical Research Centre of Finland, 2002. – 107 p.
16. Gary Chin. Agile Project Management: How to Succeed in the Face of Changing Project Requirements, by AMACOM, 2004. – 229 p.
17. Charles G. Cobb. Making Sense of Agile Project Management: Balancing Control and Agility, by Wiley, 2011. – 265 p.
18. Michael Cohn. Succeeding with Agile: Software Development Using Scrum, by Addison-Wesley Professional, 2009. – 504 p.
19. Clifford F. Gray, Erik W. Larson. Project Management: The Managerial Process, 4th Edition, by McGraw-Hill/Irwin, 2008. - 608 p.
20. Bill Holtsnider, Tom Wheeler, George Stragand and Joseph Gee. Agile Development & Business Goals, by Elsevier Inc, 2010. – 256 p.
21. Mark C. Layton. Agile Project Management for Dummies, by [John Wiley & Sons](http://82.179.249.32:2155/books.aspx?imprintid=35), 2012. – 360 p.
22. James P. Lewis. Fundamentals of Project Management (3rd Edition) by AMACOM Books, 2006. - 176 p.
23. Stanley E. Portny. Project Management For Dummies; 2nd Edition, by Willey Publishing, Inc., 2007. – 384 p.
24. Paul Roberts. Guide to Project Management, by Profile Books/The Economist, 2007. – 319 p.
25. Peter Schuh. Integrating Agile Development in the Real World, by Charles River Media, 2004. – 364 p.
26. Jason Westland, Project Management Life Cycle, by Kogan Page Ltd., 2006. – 255 p.
27. A Guide to the Project Management Body of Knowledge (PMBOK Guide), Fourth Edition, Project Management Institute, Inc., 2008.
28. H. Frank Cervone, (2011), “Understanding agile project management methods using Scrum”, OCLC Systems & Services, Vol. 27 Iss: 1 pp. 18-22
29. Dr. Alistair Cockburn, “Using Both Incremental and Iterative Development”, 2008, The Journal of Defense Software Engineering pp.27-30
30. Pete Deemer and Gabrielle Benefield. THE SCRUM PRIMER An Introduction to Agile Project Management with Scrum Version 1.04, goodagile from [www.goodagile.com](http://www.goodagile.com), 2007
31. Gabrielle Benefield. Rolling out Agile in a Large Enterprise, HICSS '08 Proceedings of the Proceedings of the 41st Annual Hawaii International Conference on System Sciences, 2008.
32. Pankaj Jalote, Aveejeet Palit, Priya Kurien, V.T. Peethamber, “Timeboxing: a process model for iterative software development”, The Journal of System and Software (2004), pp. 117-127
33. Yu Beng Leau, Wooi Khong Loo, Wai Yip Tham and Soo Fun Tan. “Software Development Life Cycle AGILE vs Traditional Approaches”, International Conference on Information and Network Technology (ICINT 2012)
34. Louise Ledbrook, “Waterfall Project Management: An Overview”, 2012, <http://projectcommunityonline.com/waterfall-project-management-an-overview.html>
35. Xavier Amatriain Rubio, Gemma Hornos Cirera. Agile Methods in Research (presentation), by Telefonica, 2008. – 42 s.
36. Ken Schwaber, Jeff Sutherland, “The Scrum Guide”, 2011. – 16 p.
37. Melonfire, “Understanding the pros and cons of the Waterfall Model of software development”, 2006, <http://www.techrepublic.com/article/understanding-the-pros-and-cons-of-the-waterfall-model-of-software-development/6118423>
38. [www.agilelogic.com](http://www.agilelogic.com)
39. <http://agilemanifesto.org>
40. [www.banki.ru](http://www.banki.ru)
41. [www.bankir.ru](http://www.bankir.ru)
42. [www.creditregistry.ru](http://www.creditregistry.ru)
43. <http://itlegalsolutions.com/e-posobie>
44. [www.koltsov.by](http://www.koltsov.by)
45. <http://rsdn.ru/>
46. [www.versionone.com](http://www.versionone.com)
47. <http://www.wisegeek.com>

# ПРИЛОЖЕНИЕ

Приложение 1. Моделирование базового плана проекта (с детализацией задач), управляемого на основании методологии «скрам».



1. Ebrary **Making Sense of Agile Project Management : Balancing Control and Agility** [↑](#footnote-ref-1)
2. http://www.banki.ru/banks/ratings/?utm\_source=google&utm\_medium=cpc&utm\_campaign=Banki\_Rossii [↑](#footnote-ref-2)
3. <http://creditregistry.ru/> [↑](#footnote-ref-3)
4. <http://agilemanifesto.org/principles.html> [↑](#footnote-ref-4)
5. 7th Annual State of Agile Development Survey <http://www.versionone.com/pdf/7th-Annual-State-of-Agile-Development-Survey.pdf> [↑](#footnote-ref-5)
6. 6th Annual State of Agile Development Survey <http://www.versionone.com/state_of_agile_development_survey/2011/> [↑](#footnote-ref-6)
7. 7th Annual State of Agile Development Survey <http://www.versionone.com/pdf/7th-Annual-State-of-Agile-Development-Survey.pdf> [↑](#footnote-ref-7)