

УПРАВЛЕНИЕ ПРОГРАММНЫМИ ПРОЕКТАМИ

УРОК № 4

ПОДРОБНЕЕ О SCRUM

СОДЕРЖАНИЕ

1. Что такое Scrum?	
Причины возникновения Scrum	4
2. Роли в Scrum	8
3. Документы в Scrum	13
Журнал продукта	13
Журнал спринта	19
Диаграмма BurnDown	20
Scrum-доска	22
4. События Scrum	24
Что такое спринт?	24
Планирование спринтов	24
Ежедневный скрам	26
Обзор спринта	27
Ретроспективное собрание	28

5. Утилиты и инструментальные средства, используемые при работе в проектах	34
Системы контроля версий.....	34
Основы работы с SVN.....	37
Основы работы с Git.....	46
6. Баг-трекеры.....	52
Основные приемы работы в Jira.....	54
Mantis BT	58
7. Системы управления проектами.....	59
MS Project.....	59
Easy Projects	71
Team Foundation Server	72
8. Домашнее задание	80

1. ЧТО ТАКОЕ SCRUM? ПРИЧИНЫ ВОЗНИKНОВЕНИЯ SCRUM

Подход Scrum был разработан для улучшения ситуации с успешностью ИТ-проектов, которая в начале 90-х годов XX века, была удручающей.

В отчете 1995 года The Chaos report, выпущенном компанией The Standish Group, приведена следующая статистика: всего 16,2% ИТ-проектов уложились в плановые сроки и бюджет, при этом 31,1% ИТ-проектов были отменены по разным причинам. По расчетам, сделанным в этом отчете, американские компании потеряли на отмененных ИТ-проектах в 1995 году более \$81 млрд и еще на \$59 млрд были превышенны первоначальные бюджеты завершенных ИТ-проектов (https://www.standishgroup.com/sample_research_files/chaos_report_1994.pdf).

Участники команд по разработке ИТ-продуктов начали рефлексировать над полученным в результате реализации проектов по водопадной модели опытом, и искать альтернативные подходы к выполнению проектов. Еще в 1986 году в HARVARD BUSINESS REVIEW вышла статья Такеучи и Нонака под названием “The New New Product Development Game”, в которой авторы описали подход «регби» (<https://hbr.org/1986/01/the-new-new-product-development-game>).

В водопадном подходе к реализации ИТ-проектов использовался последовательный подход к разработке продукта, когда команда работает как эстафетная команда,

передавая результаты работ с одного этапа на другой. В подходе, названном «регби», команда проекта старалась пройти всю дистанцию проекта вместе, как бы передавая мяч друг другу. Авторы статьи рассмотрели управление проектами в таких транснациональных компаниях, как Fuji-Xerox, Canon, Honda, NEC, Epson, Brother, 3M, Xerox и Hewlett-Packard, и проанализировали процесс разработки шести продуктов. В итоге исследования появилось описание итерационной модели разработки продуктов и шесть характеристик работы в проекте, которые, по мнению авторов статьи, помогали командам в управлении разработкой новых продуктов.

Благодаря этой статье появился термин «**подход регби**», который в 1991 году ДеГрейс и Шталь в книге «Нечестивые проблемы, праведные решения» назвали термином **Scrum**.

А в 1995 году на одной из конференций Дж. Сазерленд и К. Швабер впервые представили миру структурированный подход Scrum.

Scrum – это подход, включающий набор принципов, на которых строится процесс разработки, три роли, которые должны выполняться в проекте, пять типов событий и два документа.

Этого набора ролей, документов и событий оказалось достаточно, чтобы реализовывать ИТ-проекты и создавать в итоге хорошие ИТ-продукты.

Документ, который описывает, как работает Scrum, называется **Scrum Guide** и он доступен для бесплатного использования по ссылке: <https://www.scrumguides.org/docs/scrumguide/v2016/2016-Scrum-Guide-Russian.pdf>.

В 2001 году адепты итерационной модели жизненного цикла собрались в США, чтобы обсудить, что общего есть в их подходах. Результатом этой встречи стал Agile-манифест и 12 принципов работы по Agile.

Согласно “12th annual State of Agile report”, проведенному в 2018 году, фреймворк Scrum – самый популярный Agile-подход в мире:

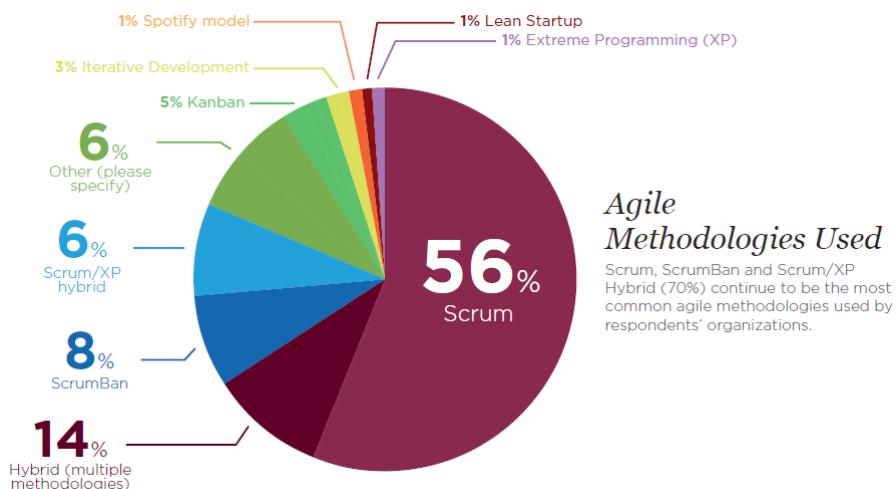


Рисунок 1

1. Что такое Scrum? Причины возникновения Scrum

Все составляющие подхода Scrum отражены на рисунке 2.

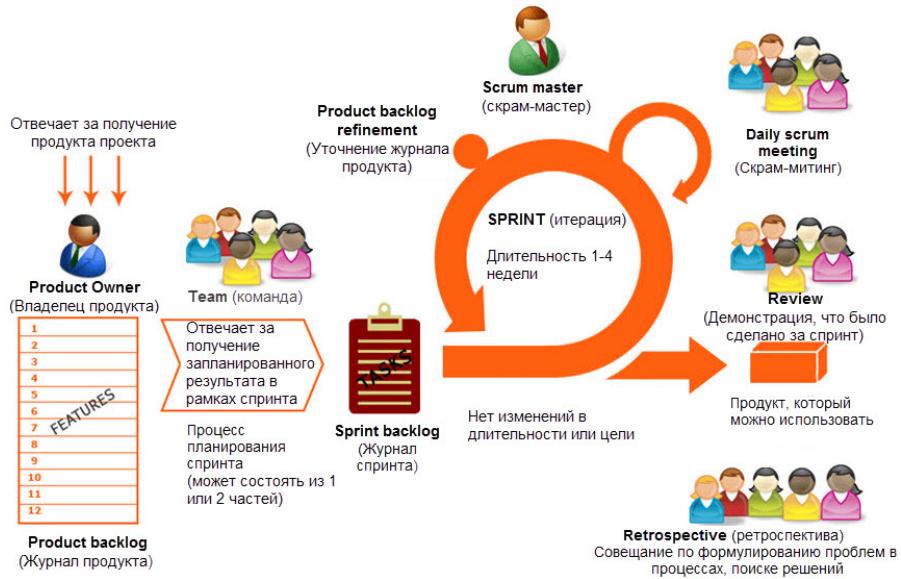


Рисунок 2

2. РОЛИ В SCRUM

Первое удивительное открытие при знакомстве со Scrum – в этом подходе отсутствует роль руководителя проекта. Многие участники тренингов по Scrum высказывают недоумение: как можно управлять проектом, если в нем нет человека, отвечающего за успех? А о том, что такое успех проекта, мы уже рассуждали в уроке 2: чаще всего об успехе проекта судят по достижению целей проекта в плановые сроки и бюджет.

В Scrum Guide описано всего три роли – **Product Owner** (владелец продукта), **Scrum Master** (скрам-мастер), **Development Team** (команда разработки продукта). Эти три роли входят в состав Scrum-команды:

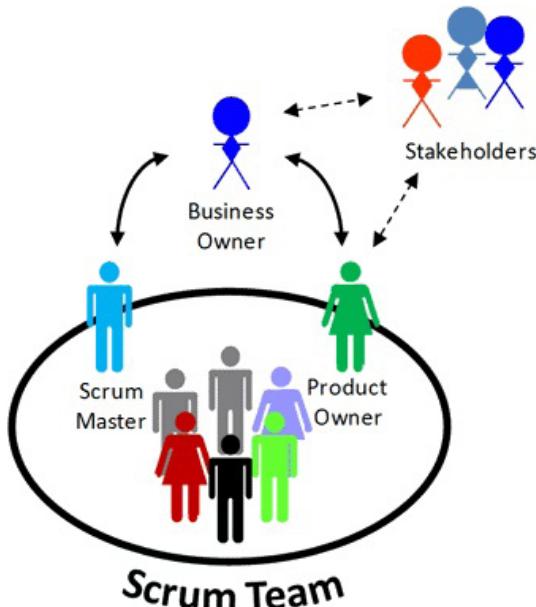


Рисунок 3

Владелец продукта отвечает за создание продукта с максимальной ценностью для потребителей этого продукта.

Владельцем продукта желательно выбрать представителя заказчика проекта, который больше всего заинтересован в получении результата от использования программного продукта, который создается в проекте.

Например, для проекта разработки интернет-банка для физических лиц, владельцем продукта может стать директор департамента розничного сектора.

Если от компании-заказчика нет человека, который способен выполнять роль владельца продукта, то команда разработки может выбрать **прокси-владельца продукта**, который будет выполнять большую часть работы за представителя заказчика, согласовывая с ним приоритеты в журнале продукта и проводя для него демонстрацию раз в месяц. Роль прокси-владельца продукта может хорошо выполнить участник, у которого есть опыт работы бизнес-аналитиком или руководителем проекта.

Для того чтобы создать правильный продукт владелец продукта использует **журнал продукта (product backlog)** и управляет содержанием этого журнала. Также владелец продукта отвечает за расходование бюджета проекта, т. к. через управление приоритетами в журнале продукта, он решает, на какие именно элементы продукта должны быть потрачены деньги проекта, а на какие из элементов денег в бюджете может не хватить.

Для расстановки приоритетов в журнале продукта владелец продукта должен разработать и внедрить подход к приоритизации. На вопрос: «А может ли быть у одного продукта несколько владельцев продуктов?»,

Scrum Guide дает отрицательный ответ. Что интересно, владелец продукта может сам исполнять некоторые элементы из журнала продукта, но чаще всего он делегирует выполнение работ команде разработки.

Задачи владельца продукта:

- фокусируется на вопросе «Что?», а не «Как?»;
- отвечает за показатели экономической эффективности проекта;
- утверждает требования к продукту;
- приоритизирует требования к продукту;
- участвует в планировании спринта;
- принимает работу спринта.

Миссия Scrum-мастера – создать результативную команду и обучить ее подходу Scrum.

Для того чтобы результативно выполнять роль Scrum-мастера, нужны навыки коучинга, фасилитации совещаний, управления групповой динамикой. Если у кандидата на эту роль нет нужных навыков, то внедрение Scrum может буксовать или вообще стать неудачным экспериментом. Поэтому на роль Scrum-мастера для команды, которая впервые работает по Scrum, лучше пригласить опытного Scrum-мастера со стороны, который обучит всех участников команды проекта, как работать в новой парадигме, и обучит одного из участников команды работе в роли Scrum-мастера.

Возможен также вариант, когда опытный участник команды разработки, ранее работавший в роли Scrum-мастера, захочет выступить в этой роли, совмещая при этом работу в роли участника команды.

Scrum-мастер может оказывать услуги владельцу продукта:

- помочь внедрить хорошие практики для расстановки приоритетов в журнале продукта;
- донести видение продукта до команды проекта;
- сформировать у команды проекта навыки по формулированию элементов в журнале продукта.

Для команды Scrum-мастер – это неформальный лидер, который помогает осознать ценности и принципы Agile, внедрить события и документы для работы по Scrum.

Scrum-мастер может оказывать для команды разработки следующие услуги:

- обучать участников команды разработки приемам самоорганизации, тайм-менеджмента, обучать оценке сложности элементов бэклога и задач;
- устранять препятствия, мешающие росту производительности в команде разработки;
- фасилитировать события Scrum.

Фасилитатор нужен для всех событий Scrum, для того чтобы они достигали поставленной цели и не превышали отведенное время. Фасилитатор направляет ход дискуссий и привносит в обсуждения четыре ценности совместной работы:

- 1) полноценное участие;
- 2) взаимопонимание;
- 3) взаимоприемлемые решения;
- 4) общая ответственность.

Команда разработки в Scrum должна нести ответственность за выполнение всех запланированных на спринт задач. Для того чтобы команда разработки была результативной, она должна соответствовать следующим характеристикам.

В команде разработки нет никаких других ролей, кроме как роль разработчика.

Команда должна быть кроссфункциональной, т. е. в ней должны быть представлены все специализации, которые нужны для получения продукта проекта. Внутри команды могут быть специализации труда, но при этом ответственность за результат лежит на команде. Многие команды боксуют, когда специалисты внутри команды имеют узкую специализацию.

Команда разработки должна быть самоорганизующейся. Это значит Scrum-мастер и владелец продукта не должны диктовать команде, как выполнять работу и определять, кто будет исполнителем задач в рамках спринта. Практикуется самостоятельный выбор каждым участником тех задач, над которыми он будет работать в спринте.

У команды разработки не должно быть никаких подкоманд. При этом размер команды не должен превышать девяти человек.

Команда разработки должна нести коллективную ответственность за результат спринта.

3. ДОКУМЕНТЫ В SCRUM

Фреймворк Scrum был разработан так, чтобы минимизировать количество документов, необходимых для достижения цели проекта. Таким образом, в фреймворке предлагается использовать только два документа:

1. Журнал продукта (**Product backlog**).
2. Журнал спринга (**Sprint backlog**).

Журнал продукта

Product backlog – это документ, который должен включать все элементы работ, которые должны быть выполнены командой разработки. В журнал попадают также сообщения об ошибках, идеи о новых функциях и т. д.

Требования к продукту в Scrum бывают двух типов: **epic** – требования, реализация которых не может быть выполнена за один спринт, и **story** (*история пользователя*) – требование, которое можно реализовать за один спринт.

Описание истории делается в таком формате:

Будучи <тип пользователя>, я хочу <конкретная цель>, чтобы <конкретная причина>.

Как создавать журнал продукта и управлять его развитием

Работа с требованиями в Scrum ведется по большей части в устном виде, чем в письменном. И хотя журнал продукта содержит письменные требования, но вместо

того, чтобы описывать каждое требование в виде подробного документа (типа SRS), команда описывает его одним предложением. Как только история пользователя поднимается в рейтинге журнала продукта настолько высоко, что с высокой вероятностью она может попасть в ближайший спринт, команда разработки решает на одном из планирований спринтов, кто возьмет историю в проработку; и исполнитель этой задачи уточняет вопросы по истории, к моменту планирования данной истории в реализацию, для следующего спрингта.

Журнал продукта не бывает исчерпывающим, а начальный вариант журнала содержит только известные на момент старта проекта и наиболее понятные требования. Поэтому для создания первой версии журнала продукта команде разработки и владельцу продукта достаточно извлечь бизнес-требования к продукту, отранжировать их, и проработать самые важные из них до уровня пользовательских требований (по классификации Вигерса).

Рассмотрим, какими характеристиками должен обладать хороший журнал продукта.

- **Оптимальный уровень детализации историй пользователей.**

Истории, находящиеся в журнале продукта, которые попадут в реализацию в один из ближайших спринтов, должны быть настолько хорошо описаны и оценены, чтобы их можно было реализовать в течение предстоящего спрингта. Истории, которые попадут в реализацию более чем через три спрингта, должны быть описаны менее подробно.

- **Наличие предварительных оценок трудоемкости.**

Элементы журнала (истории, эпики, описания ошибок), находящиеся в нижней части журнала продукта, как правило, еще не совсем проработаны аналитиком. Поэтому связанные с ними оценки могут быть менее точными, чем оценки, данные элементам в верхней части журнала.

- **Изменчивость.**

Журнал продукта по мере развития продукта проекта изменяется: в нем появляются новые истории пользователей, какие-то из историй удаляются, изменяются приоритеты историй.

- **Приоритизация журнала.**

Журнал продукта должен быть отсортирован таким образом, чтобы самые важные элементы находились вверху списка, а наименее важные – внизу. Предполагается, что при правильной расстановке владельцем продукта приоритетов, команда может максимизировать ценность разрабатываемого продукта или системы.

Извлечение требований и их анализ может осуществлять как владелец продукта, так и команда, а вот за наличие приоритетов у элементов журнала продукта отвечает владелец продукта.

Scrum-команда решает сама, как и когда должно производиться уточнение журнала продукта с целью подготовки историй для следующего планирования спринта: уточнения требований к историям и другим элементам журнала и их предварительной оценки. Эта деятельность обычно занимает не более 10% от доступного времени Scrum-команды.

Структура журнала продукта может быть такой:

Название	Важность	Оценка	Как продемонстрировать	Примечание
Как пользователь автомобиля, я хочу иметь возможность прослушивать мои письма из указанного e-mail за рулем	100	40	Водитель заводит двигатель, нажимает кнопку e-mail на дисплее, затем кнопку «Прослушать e-mail». Система зачитывает сообщения из настроенного ранее ящика e-mail	О возможности указывать не сколько e-mail ящиков пока не стоит беспокоиться
Как пользователь автомобиля, я хочу, чтобы система подсказывала мне, когда я меняю полосу движения (чтобы я не смог заснуть за рулем)	99	50	Водитель включает на дисплее опцию «Оповещать о смене полосы», при движении по шоссе он поворачивает руль и автомобиль меняет полосу. Система голосом оповещает: «Вы изменили полосу движения»	

Поле «Важность» заполняет владелец продукта, данные по оценке сложности элемента предоставляет команда разработки, и внесенная в журнал продукта оценка считается предварительной, т. к. при планировании спринта оценка может быть уточнена.

Журнал продукта существует до тех пор, пока продукт проекта не будет готов и принят владельцем продукта.

Если над одним продуктом работают несколько команд, то для описания всех предстоящих работ используется один журнал продукта.

Как оценивать задачи в журнале продукта (product backlog)?

Существует несколько способов для предварительной оценки элементов журнала продукта в Scrum. Один из них – **покерное планирование**.

Команда разработки, Scrum-мастер и владелец продукта собираются в одном помещении, владелец продукта знакомит команду с историями, получившими самый высокий приоритет в журнале продукта. У каждого участника команды есть колода специальных карт. Можно использовать колоду карт для покер-плэннинга, в которой значения некоторых цифр на картах подобраны по ряду Фибоначчи.



Рисунок 4

В этой колоде карт не все цифры на картах соответствуют ряду Фибоначчи: например, 1/2, 20, 40 и 100, т. к. в ряде Фибоначчи сумма двух предыдущих чисел должна давать следующее число. В колоде есть карта «?», которая означает, что участник оценки не понимает суть истории и не готов сделать оценку, а карта «кофейная чашка» означает, что участник устал и предлагает сделать перерыв, карта «0» означает, что этот элемент уже реализован.

Владелец продукта знакомит команду с историей, которая получила самый высокий приоритет. Участники команды задают ему вопросы, и он отвечает на них. После этого, участники команды делят историю на задачи и по каждой задаче происходит оценка трудозатрат. При этом участники заранее договариваются о том, в каких единицах будут делать оценки трудоемкости задач: в **идеальных часах** или в **story points**.

Story points – это относительные оценки объема работы для истории. Команда при оценке сравнивает данную историю с эталонной историей, которая была оценена в 1 story point. Если данная история сложнее в три раза, чем эталонная, то она получит оценку в три story points.

Каждый участник думает о том, сколько лично у него ушло бы времени на реализацию задачи и, сделав оценку в голове, он должен вытянуть карту с цифрой, соответствующей этой оценке. При этом, для того чтобы избежать «эффекта привязки», участники оценки кладут карты рубашкой вверх. Например, один участник оценивает выполнение задачи в 4 человека-часа. В этом случае он

вытягивает из колоды 2 карты: с цифрой 3 и с цифрой 1 и кладет их рубашкой вверх.

После того как все участники вытянули карты, карты вскрываются и участники оценки видят результаты.

Цель покерного планирования – сблизить мнение участников, поэтому владелец продукта просит высказаться участника, у которого получилась самая низкая оценка и участника, у которого оказалась самая высокая оценка. После высказываний этих участников, как правило, уточняются требования к задаче, выбирается самый лучший подход к реализации задачи и происходит второй тур оценки. После второго тура диапазон оценок становится более узким. Для перехода к одноточечной оценке можно вычислить среднее арифметическое и перейти к оценке следующей задачи.

Так команда оценивает все истории и задачи, которые их просит оценить владелец продукта.

Журнал спринга (sprint backlog)

Документ содержит задачи и истории пользователей, которые команда отобрала для реализации в следующем спринте. Для того чтобы отобрать истории и задачи на спринт, команде надо знать приоритет историй в журнале продукта, емкость спринга, и предварительные оценки элементов журнала продукта.

Емкость спринга – это количество времени, которое есть у всех участников команды на следующий спринт. Рассчитывает емкость спринга Scrum-мастер, по итогам опроса участников команды разработки.

Структура журнала спринга может иметь такой вид:

Задачи	Оценка	Исполнители	СДЕЛАНО?
Разработать документ по сервису (аналог Заказ-наряда в старой системе). Прикрутить к Заказ-наряду плановое поступление денежных средств, использовать справочник физлиц	8	Сергей	нет
Загрузить данные по номенклатуре из Excel в рабочую базу	2	Максим	да
УстраниТЬ баг с созданием счета на основании Заказа клиента в бизнес-процессе	2	Ирина	да

Диаграмма BurnDown

Для понимания того, успеет ли команда реализовать взятые на спринт обязательства, нужно знать, какой объем работ реально остался до окончания спринга и какой должен был остаться по плановым оценкам.

Для этого и используется **диаграмма BurnDown**, пример которой представлен на рисунке 5.

У команды на спринт было запланировано работы на 162 идеальных человека-часа. Зеленая линия на диаграмме показывает, какой объем работ в чел.-часах должен оставаться у команды в идеальной ситуации на каждый день. Красная линия показывает, какой объем работ реально у команды остался. Таким образом, если красная линия идет выше зеленоЙ – мы отстаем от плана, в обратном случае – мы его опережаем или идем ровно так, как запланировали (если линии совпадают).

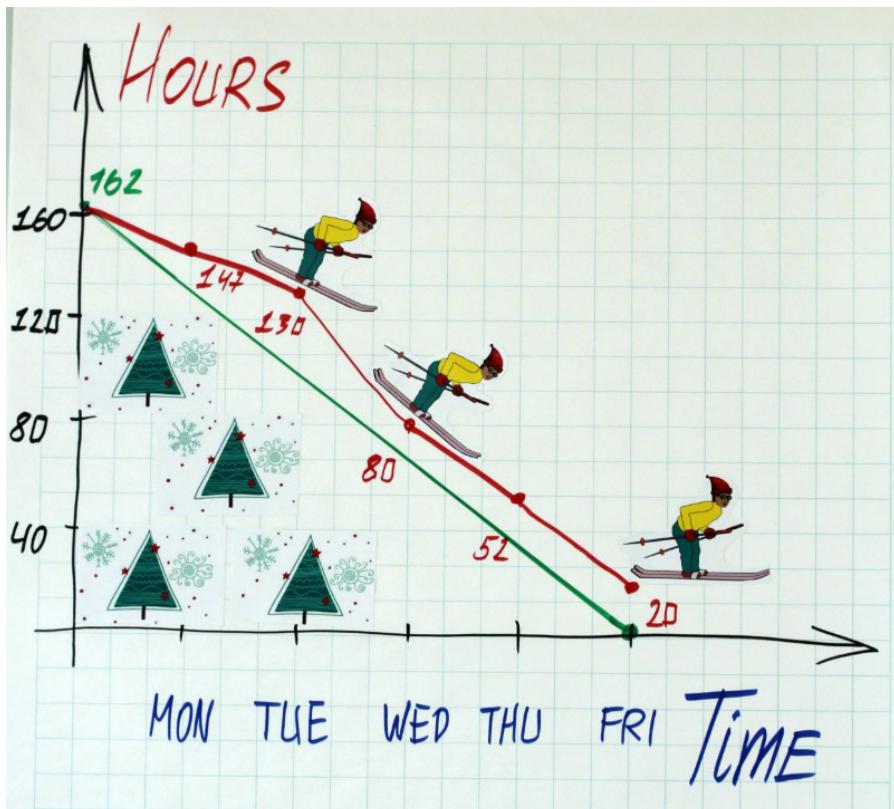


Рисунок 5

Если участники команды не ленятся, и каждый день оценивают по уже начатым задачам объем оставшихся работ, команда видит на диаграмме прогноз относительно того, успевает ли она в срок выполнить все намеченное на спринт.

Scrum-доска

Для визуализации потока задач и контроля задач в спринте команда может использовать **Scrum-доску**, пример которой показан ниже:

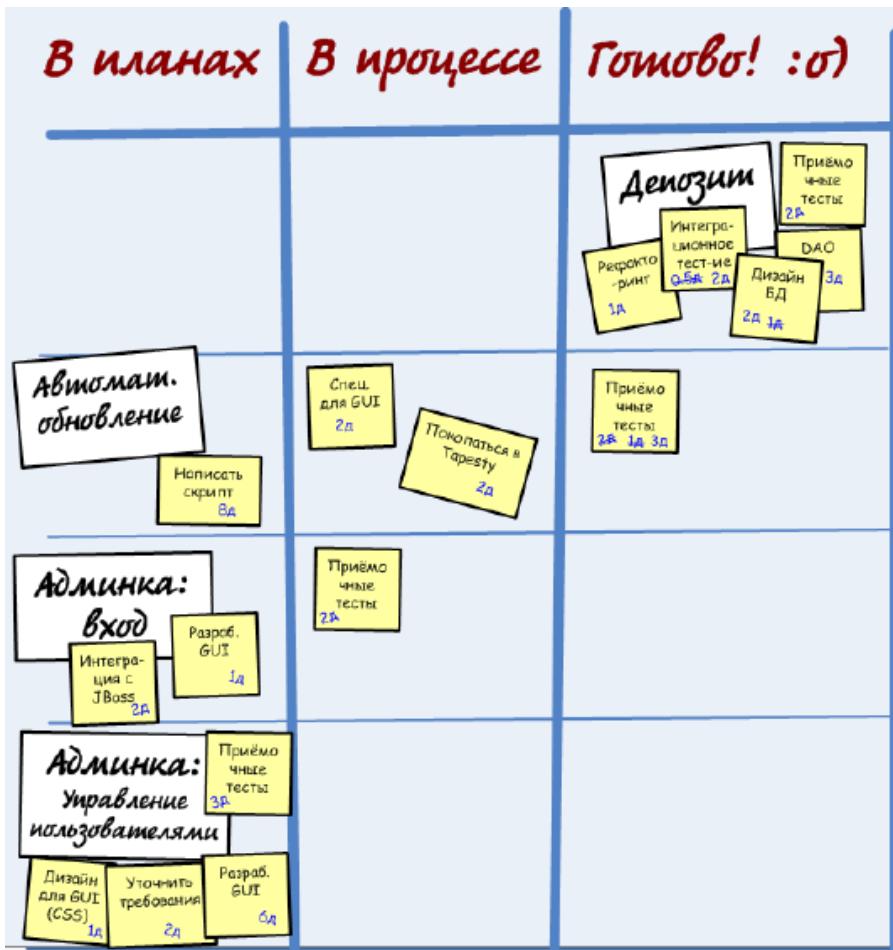


Рисунок 6

Все истории, попавшие в спринт, описаны на белых стикерах, а на желтых стикерах – задачи, на которые разбита эта история:



Рисунок 7

Под каждую историю на доске создана своя дорожка, по которой перемещаются слева направо задачи этой истории, описанные на желтых бумажках.

На доске есть три столбца, описывающие состояния задач: «В планах», «В процессе» и «Готово». Как только один из участников команды взял задачу в работу, он может переместить стикер с этой задачей в столбец «В работе». Как только задача завершена – стикер перемещается в столбец «Готово». И так все стикеры постепенно передвигаются в столбец «Готово». Рядом с этими столбцами можно поместить BurnDown диаграмму.

4. СОБЫТИЯ SCRUM

Что такое спринт?

В Scrum используется термин «спринт» для обозначения отрезка времени длительностью от одной до четырех недель, в рамках которого должно быть созданы дополнительные функции продукта:

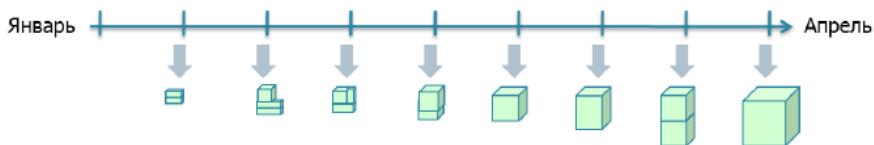


Рисунок 8

Длительность спринтов определяется владельцем продукта совместно с командой и не изменяется по ходу проекта.

Планирование спринтов

Результатом планирования спринта должен стать список задач, которые команда обязуется реализовать за спринт. Этот список задач помещается в журнал спринта. Как правило, планирование спринта состоит из двух этапов:

- На первом этапе команда готовится к планированию спринта:** эпики разбиваются на истории, истории разбиваются на задачи и оцениваются.

- 2. На втором этапе проводится митинг по планированию спрингта**, на котором команда определяет, сколько работы она сможет реализовать за спринт. Для этого команде нужно понимать емкость спрингта – количество человеко-часов, которое есть у всех участников команды на следующий спринт, и предварительные оценки сложности задач.

Рекомендуется митинг по планированию спрингта проводить за 1-2 часа.

Фасилитирует этот митинг скрам-мастер, но он может предложить это сделать любому участнику команды разработки для того, чтобы у него, со временем, сформировать компетенцию по фасилитации митингов и создать условия для взаимозаменяемости участников команды проекта.

Владелец продукта обязательно присутствует на планировании спрингта для того, чтобы:

- отвечать на вопросы команды по поводу требований к истории или эпике;
- принимать решения о том, что делать в ситуации, когда один из элементов журнала не помещается в спринт, но владельцу продукта очень надо, чтобы была сделана хоть какая-то работа по этому элементу в следующем спринте;
- задавать вопросы участникам команды разработки по поводу того, почему они именно так оценивают сложность задач, если есть сомнения в адекватности оценки.

Ежедневный скрам (daily scrum)

Этот митинг проводится для того, чтобы решить следующие проблемы:

- снизить действие студенческого синдрома, которому подвержено большинство людей;

Студенческий синдром – это откладывание выполнение задачи до последнего момента; интенсивность работы над задачей описывается кривой:

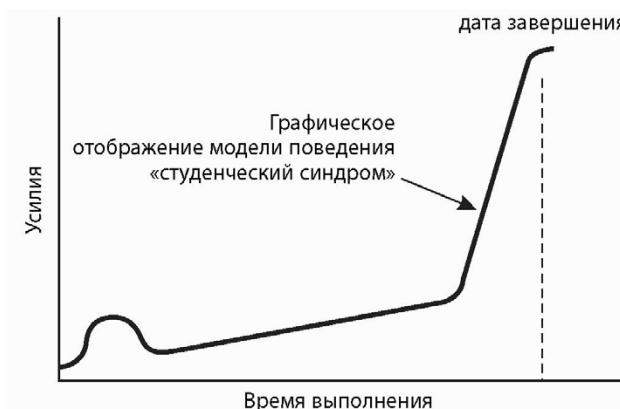


Рисунок 9

- помочь исполнителю продвинуться в решении сложности с получением результата, которую он не стал обсуждать с коллегами, а попытался решать самостоятельно.

Каждый участник команды разработки рассказывает коллегам, что он успел сделать вчера и что ему помешало реализовать запланированный объем работ и получить ожидаемый результат. Многих из нас мотивирует к действию осознание того, что если я сегодня не закончу хотя

бы одну задачу, то завтра, на ежедневном скраме, мне нечего будет рассказать коллегам.

Ежедневный Scrum-митинг проводится каждый день и в нем участвует каждый участник команды, который отвечает на три вопроса:

1. Что я сделал вчера?
2. Что мне мешало сделать запланированный объем работ на вчера?
3. Что я буду делать сегодня?

Для того чтобы митинг прошел за 15 минут, его проводят стоя. Скрам-мастер следит за временем митинга, и за тем, чтобы все проблемы не обсуждались прямо во время митинга, а выносились на отдельные совещания. Он предлагает провести совещания по решению проблем, на которые приглашаются только те участники команды, которым интересен обсуждаемый вопрос и кто может помочь.

Обзор спринта

Для того чтобы получить быструю обратную связь по сделанным функциям от владельца продукта проводится **демонстрация** (коротко – **демо**) того, что было сделано командой за спринт.

На демо приходит вся команда, каждый участник должен показать владельцу продукта результаты, который может по итогу увиденного уточнить требования или у него могут появиться идеи по новым требованиям.

Команда не тратит много времени на разработку ТЗ, а работает непосредственно над продуктом, создавая

быстрые прототипы, при этом ей приходится тратить время на сбор обратной связи. Желательно ограничить демо во времени, проводя его не более чем за 4 часа в случае 4-недельных спринтов, и не более 1 часа, в случае 1-недельных спринтов.

Во время проведения демо нужно, чтобы кто-то из участников команды записывал полученную обратную связь от владельца продукта. В конце демо следует продемонстрировать все сделанные записи владельцу продукта и получить подтверждение, что все записано верно.

По итогам демо владелец продукта добавляет новые пожелания в журнал продукта и приоритизирует их.

Ретроспективное собрание

В Scrum заложена идея непрерывного улучшения процесса командной работы по Scrum.

Ретроспектизы следует проводить по окончании каждого спринта и ограничить их по времени – не более чем 4-мя часами.

В теории решения изобретательских задач (ТРИЗ) описан **Закон повышения идеальности системы**: развитие технической системы должно приводить к увеличению степени ее идеальности.

Для оценки идеальности системы была предложена формула:

$$\text{Идеальность системы} = \frac{\text{Ценности (Values)} - \text{Потери качества}}{\text{Затраты}}$$

Система тем более идеальна, чем больше она создает ценности, порождая при этом минимальные затраты и сводя к минимуму потери качества.

Из этой формулы вытекает парадоксальное, на первый взгляд, следствие: идеальная система – это система, которая не порождает затрат, но при этом она создает полезность.

Забота о повышении идеальности процесса Scrum лежит на Scrum-мастере. Для этого он организует и проводит совещания, которые называются «ретроспективами».

На классических ретроспективах обсуждают три вопроса:

1. Что у нас работает хорошо?
2. Что могло бы работать лучше?
3. Что из этого нам необходимо улучшать?

Список вопросов для ретроспективы можно расширить, например так:

1. Обсуждение выполнения задач, которые были поставлены участникам команды на предыдущей ретроспективе, а именно:
 - Какие задачи по улучшению нашего процесса выполнены?
 - Какие задачи не удалось выполнить и по каким причинам?
 - Каковы новые сроки реализации по невыполненным задачам?
2. Обсуждение фактических значений метрик команды, например velocity и focus factor.

3. Что помешало команде получить лучшие значения по метрикам?
4. Какие новые задачи по улучшению процесса мы поставим на следующий спринт? Ответственные и сроки по задачам, связанным с улучшениями?

Для проведения нескучных ретроспектив нужно использовать специальные игры, которые повышают вовлеченность сотрудников, например: «Линия времени», «Цветные точки», «Раздраженный, грустный, довольный», упражнение «Подобное к подобному» (*подробнее смотри в книге «Agile-ретроспектива. Как превратить хорошую команду в великую».*)

Пример разработки продукта по Scrum представлен ниже.

1. Команда разработки вместе с владельцем продукта на установочном совещании по проекту определили, что длительность спринта будет равно одной неделе.
2. Перед началом работы владелец продукта разработал первую версию журнала продукта и включил в него 20 историй, каждая история получила значение в поле «Важность» и описание в поле «Как продемонстрировать». После чего владелец продукта попросил команду разработки сделать предварительные оценки сложности. Команда для оценки сложности историй выбрала метрику – идеальные чело-веко-часы. Один из участников команды предложил свою помочь в оценке, оценил самые важные истории и заполнил поле «Оценка» (смотри пример журнала спринта выше).

3. Первое планирование спринта решили провести в понедельник в 10.00. На планировании спринта определили, что у команды есть на этой неделе 100 человеко-часов на спринт. После этого разбили истории на задачи и уточнили оценки сложности задач. Когда общая трудоемкость выбранных в спринт задач превысила 100 чел.-часов, владелец продукта предложил остановить планирование.

Команда создала журнал спринта, и участники сами распределили задачи спринта между собой. Scrum-мастер предложил проводить ежедневный скрам каждый день в 10.00, а демо спринта и ретроспективу – в пятницу с 14.00 до 18.00. Участники команды и владелец продукта согласились с предложением.

4. После планирования спринта участники разошлись на рабочие места и стали работать над задачами спринта.
5. На следующий день в 10.00 все собрались в одном помещении для проведения ежедневного скрама. Скрям-мастер предложил начать отвечать на вопросы любому желающему, первой решила выступить Ирина.

Скрям-мастер: *Что ты сделала вчера?*

Ирина: *Я закончила задачу «Устранить баг с созданием счета на основании Заказа клиента в бизнес-процессе».*

Скрям-мастер: *Что тебе помешало закончить еще одну задачу?*

Ирина: *Я не совсем понимаю, как ее реализовать?*

Скрям-мастер: *Кто может помочь Ирине?*

Костя: *У меня есть идея, как это сделать.*

Скрям-мастер: *Ирина, Костя, останьтесь после митинга и обсудите как можно решить проблему, ок?*

Коллеги: *Ок.*

Каждый ответил на три вопроса и за 15 минут митинг завершился. После митинга остались Костя и Ирина, чтобы найти решение обозначенной на митинге проблеме.

6. Каждый день на ежедневном скраме команда обнаруживала, у кого какие есть проблемы, и после митинга участники находили решения этих проблем.
7. В пятницу утром скрам-мастер подошел к каждому участнику команды и задал вопрос: «Ты знаешь, что и как будешь показывать на демо?». Если кто-то сомневался, то скрам-мастер помогал подготовиться к демо.
8. В пятницу в 14.00 в заранее забронированной скрам-мастером комнате прошло демо спринта, на котором владелец продукта дал 5 замечаний по сделанным задачам и внес 2 идеи по тому, как можно изменить созданную функциональность. Скрам-мастер записал все идеи и замечания и передал владельцу продукта. По окончании демо, владелец продукта внес все замечания и идеи в журнал продукта в течение 15 минут и проранжировал их. Демо прошло за 1 час.
9. Скрам-мастер предложил сделать перерыв в 20 минут на кофе и чай.
10. В 15.20 команда снова собралась в этом же помещении, но уже без владельца продукта.

Скрам-мастер объявил цель ретроспективы: Мы должны обнаружить приемы и инструменты, которые помогли нам в реализации этого спринта и выявить узкие места в нашем процессе для того, чтобы его улучшить. У нас есть 1 час на обсуждение этих вопросов, выступить должен каждый. Начнем с Ирины.

Скрам-мастер: Чему позитивного произошло в этом спринте?

Ирина: Я закончила все свои задачи.

Скрам-мастер: Это прекрасная новость, запишу это в раздел «Что было хорошего». А что могло бы быть лучше?

Ирина: Мы не успели реализовать все запланированные задачи.

Скрам-мастер: А что нам помешало?

Ирина: Видимо, мы не достаточно умеем планировать свой рабочий день и организовывать выполнение задач в течение дня.

Скрам-мастер: Кто согласен с этим?

(подняли руки все)

Скрам-мастер: Как мы можем изменить ситуацию?

Костя: Я читал отличную книгу по тайм-менеджменту, она сильно помогла мне. Может нам провести тренинг на эту тему?

Скрам-мастер: Кто согласен с этим предложением?

(все подняли руки)

Скрам-мастер: Отлично, я найду тренера и вышлю вам программу до конца этого спринта.

После этого скрам-мастер записал на доску в раздел «Что мы будем изменять?» задачу для себя: «Найти тренинг, получить программу и выслать коллегам на согласование. Срок: до конца спринта. Исполнитель: скрам-мастер».

Ретроспектива прошла за 1 час 15 минут.

11. Следующее планирование спринта состоялось в понедельник в 10.00.
12. Примерно таким же образом команда выполняли каждый спринт.

5. УТИЛИТЫ И ИНСТРУМЕНТАЛЬНЫЕ СРЕДСТВА, ИСПОЛЬЗУЕМЫЕ ПРИ РАБОТЕ В ПРОЕКТАХ

Системы контроля версий

При общей работе команды разработки над ИТ-проектом, возникает необходимость совместно редактировать версии документов по проекту, обеспечить хранение исходных кодов разрабатываемой программы. Для совместной работы с документами и исходным кодом можно, конечно, использовать папки на сервере, но со временем количество таких папок может значительно возрасти, что создает трудности в вопросе отката на предыдущие версии, отслеживании изменений и т. п. Для решения этой проблемы предназначены системы контроля версий.

Система контроля версий (СКВ) позволяет сохранять старые версии файлов и, в случае необходимости, возвращаться к ним; дает возможность просматривать изменения, которые внесли в предыдущие версии файла, идентифицировать автора изменений, определить, кто и когда мог внести в код ошибку и т. д. Если вы сохраните ненужные изменения в файл или потеряете его, при использовании СКВ, все можно будет легко восстановить.

Один из вариантов СКВ – это **централизованная система контроля версий (ЦСКВ)**. Такие системы включают в себя центральный сервер, на котором хранятся все

файлы под версионным контролем, и клиентские машины, которые получают копии файлов от него.

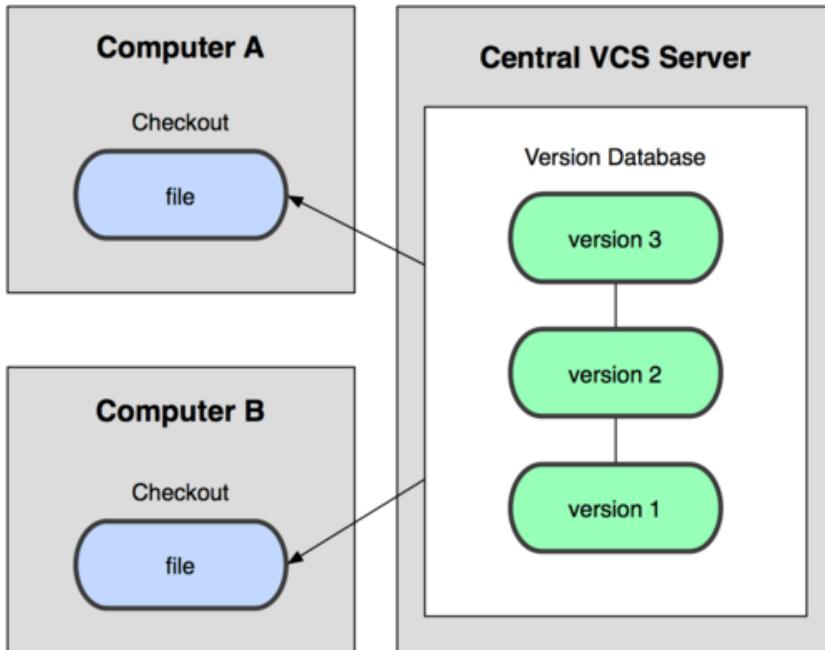


Рисунок 10

К этой категории СКВ относятся такие продукты, как **CVS** и **Subversion** (также известная как **SVN**).

Система одновременных версий (CVS) разработана в 80-х годах.

CVS позволяет делать так называемый **check-out** (**извлечение**) – получать с сервера версию файла, а затем **check-in** (**возврат**) – с внесенными изменениями.

CVS была разработана для целей совместной работы, чтобы избежать конфликта версий. Для совместной работы над проектом участникам предоставляется только самая последняя версия кода. Одним из недостатков

первых версий системы CVS было то, что пользователю необходимо было быстро зафиксировать изменения в репозитории, чтобы другие пользователи не опередили его. В одной из версий CVS получило возможность работы над проектами с ветками кода, что позволило иметь несколько вариантов продукта с разными функциями, которые можно объединить. Другими недостатками CVS считаются:

- переименование или перемещение файлов не отражается в истории;
- есть проблемы безопасности, связанные с символическими ссылками на файлы;
- нет поддержки атомарных операций, что может привести к повреждению кода;
- операции с ветками программного кода долгостоящие, так как эта система контроля не предназначена для долгосрочных проектов с ветками кода.

CVS – проверенная временем система контроля версий, которая предоставляется бесплатно, но для устранения ее недостатков на смену CVS пришла система **Subversion (SVN)**.

Разработка SVN началась в 2000 году, когда CVS уже была признана среди разработчиков программного обеспечения. Создатели SVN решили использовать уже проверенные временем концепции и при этом обеспечить легкий переход на новый продукт сторонникам CVS.

В целом, SVN превосходит CVS по всем параметрам, кроме поддержки меток, адресующих объекты файловой системы.

Недостатки SVN:

- встречаются ошибки, связанные с переименованием файлов и директорий;
- слабый набор команд для работы с репозиторием;
- относительно небольшая скорость.

Основы работы с SVN

Работу с репозиторием SVN рассмотрим на примере **TortoiseSVN**.

TortoiseSVN – это бесплатный клиент для системы контроля версий Subversion, выполненный как расширение оболочки Windows.

Репозиторий (repository) – централизованное хранилище исходных кодов, рабочих материалов и документации. Любое количество клиентов подключается к хранилищу и читает или записывает эти файлы.

В репозитории хранятся все структуры папок и файлов. Репозиторий хранит все изменения, зафиксированные в нем, с момента создания. Для отслеживания изменений во времени каждой операции, которая изменяет содержимое репозитория, ставится в соответствие уникальный «номер ревизии», запоминается время фиксации и ее автор. Все ревизии папок и файлов в репозитории доступны любому пользователю.

Для работы с репозиторием используется **браузер репозитория (repo-browser)**. Чтобы им воспользоваться, нужно войти в корневую папку любого жесткого диска

Урок №4

и по нажатию правой кнопки мыши запустить браузер:

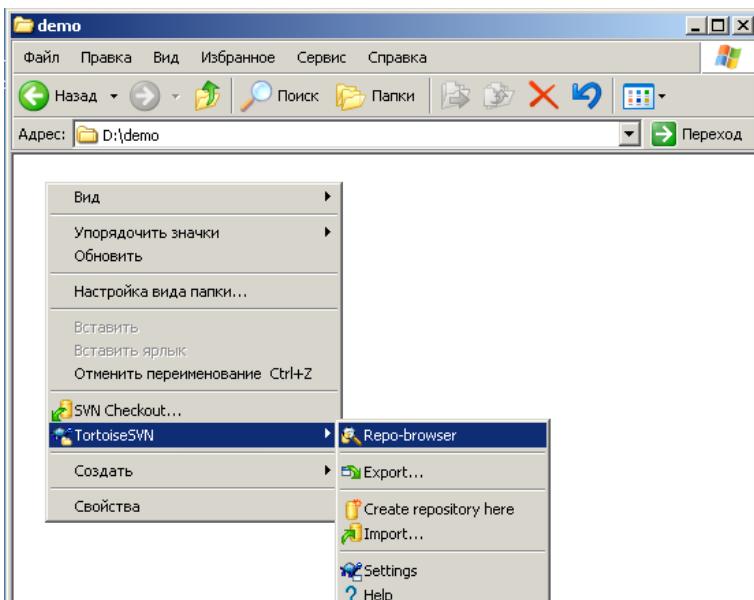


Рисунок 11

Для просмотра репозитория нужно выбрать, к какому именно репозиторию будет обращение:

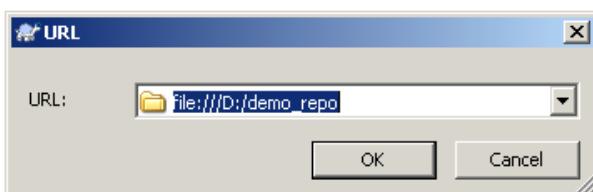


Рисунок 12

После создания репозиторий будет пустым:



Рисунок 13

Чтобы поместить свои папки и файлы под контроль SVN, необходимо создать первоначальный проект. Для этого нужно в корневой папке репозитория использовать команду **Create folder**:

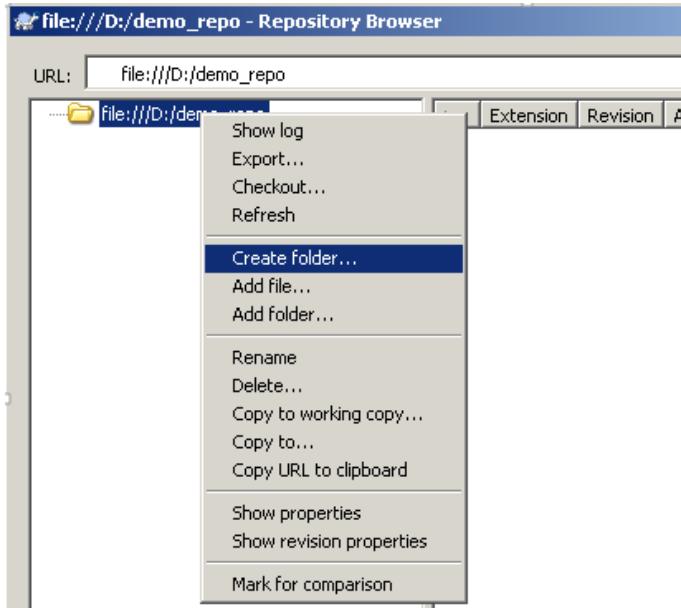


Рисунок 14

После чего, нужно ввести имя папки проекта, например **demo_project**:

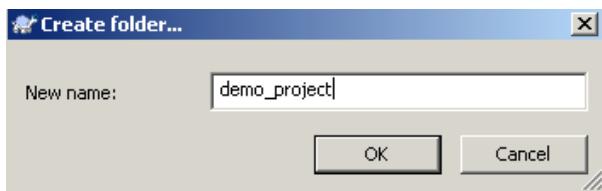


Рисунок 15

Точно так же делаем создание обязательных папок, назначение которых описано ниже, в итоге получим пустой проект в репозитории:

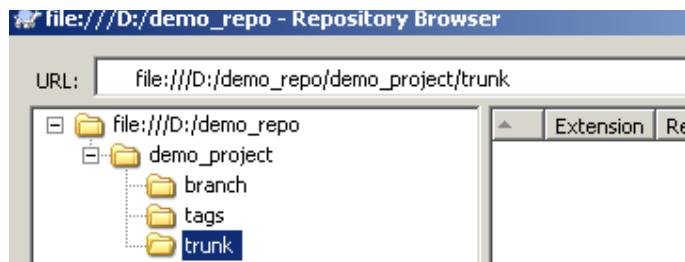


Рисунок 16

Для того чтобы начать работу с проектом, нужно создать рабочую копию. Для этого создаем корневую папку проекта на жестком диске. Заходим в эту папку и используем команду **Checkout**.

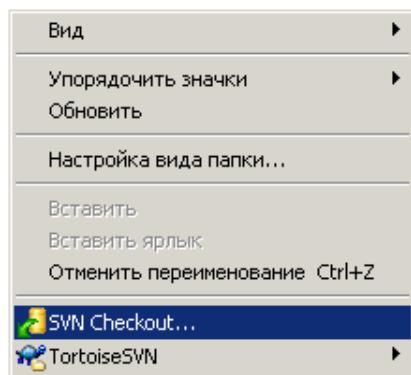
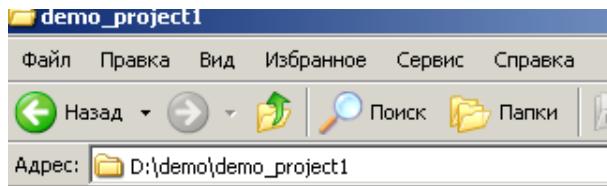


Рисунок 17

С помощью браузера репозитория выбираем интересующий нас проект, папки и файлы в этом проекте. С помощью браузера ревизий **Show log** выбираем нужный номер ревизии.

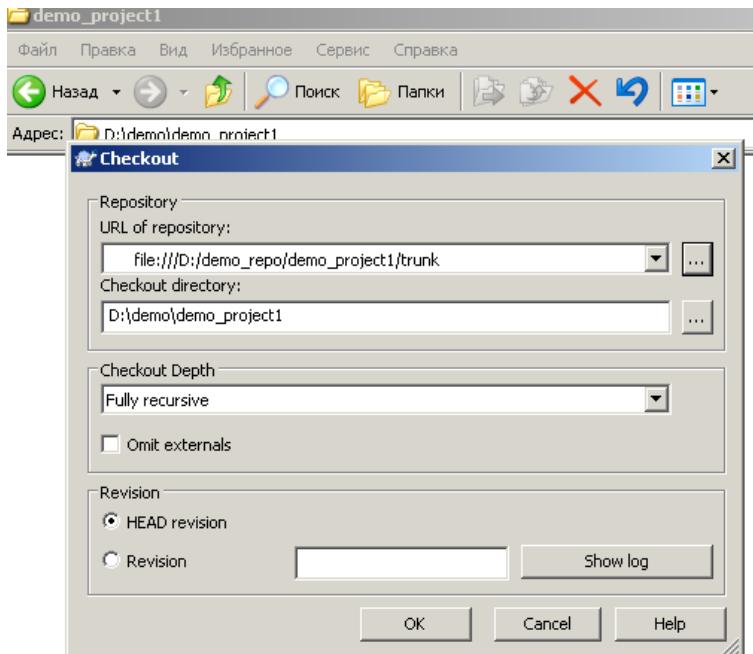


Рисунок 18

Итак, рабочая версия создана и находится под контролем SVN.

Теперь можно свободно изменять, удалять модифицировать папки и файлы проекта, добавлять новые папки и файлы не опасаясь того, что изменения будут мешать работе коллег.

Работа с рабочей копией

1. Добавление файлов в рабочую копию.

Для того чтобы поместить файл под контроль SVN, используется команда **Add**:

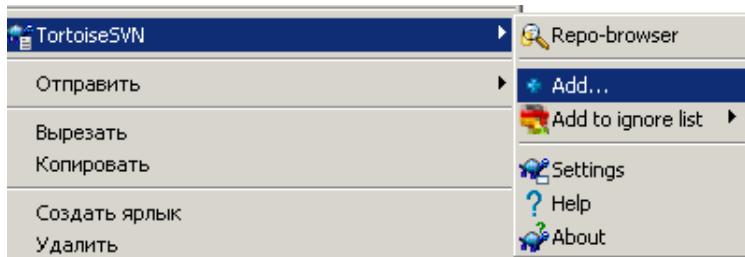


Рисунок 19

Перед выполнением команды, SVN предоставит возможность проверить список добавляемых файлов. В том случае если файл добавляется ошибочно, его добавление можно исключить.

Для фиксации изменений необходимо выполнить команду **Commit**.

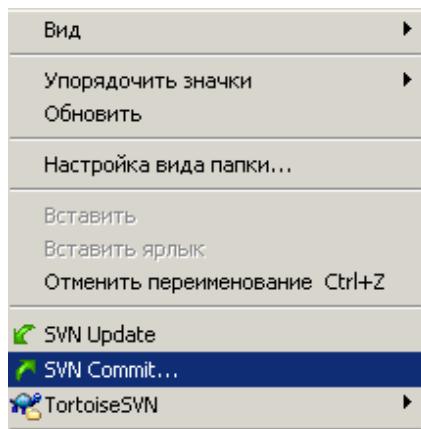


Рисунок 20

2. Синхронизация рабочей копии с репозиторием.

Допустим, два программиста независимо друг от друга добавили свои файлы в проект. Для того чтобы воспользоваться результатами работы друг друга, они должны синхронизировать свои рабочие копии. В SVN это делается через репозиторий, с помощью команды **Update**.

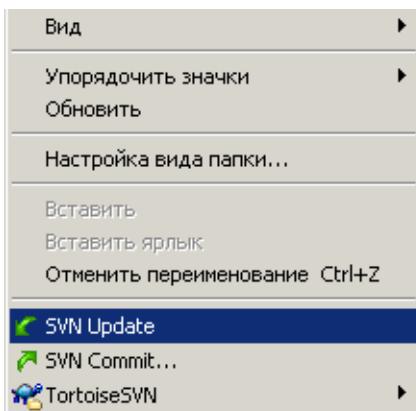


Рисунок 21

3. Изменение и откат файлов.

Для того чтобы убрать неправильные изменения из файла, используется команда **Diff** над нужным файлом.

Для того чтобы восстановить начальное состояние файла (сделать откат по ревизии) нужно воспользоваться командой **Revert**.



Рисунок 22

4. Переименование файлов.

Если файл нужно переименовать, то используется команда [Rename](#).

А чтобы отменить переименование, нужно выполнить команду [Revert](#).

5. Разрешение конфликтов.

Для разрешения конфликтов используется команда [Edit Conflicts](#).

Чтобы конфликт окончательно считался разрешенным, нужно удалить все файлы, автоматически созданные для разрешения конфликта. Это можно сделать средствами операционной системы, но лучше всего это делать с помощью команды [Resolved](#).

6. Блокировка папок и файлов.

Блокировку использовать следует в том случае, когда нужно, чтобы папки и файлы были доступны для изменения только одному человеку. Для этого нужно использовать команду [Getlock](#).

Рассмотренных команд хватит для начала работы в SVN, остальные команды нужно осваивать по мере возникновения необходимости.

У централизованных систем контроля версий есть очевидный недостаток – в случае если централизованный сервер не доступен, то в это время разработчики не могут сохранить результаты своей работы в новой версии файла.

Распределенные системы контроля версий (РСКВ) лишены этого недостатка, т. к. клиенты полностью копируют весь репозиторий. Получается, что если сервер

«ложится», то любой репозиторий от клиента может быть скопирован обратно на сервер и можно восстановить базу данных. То есть, когда клиент копирует себе свежую версию файлов, он создает себе полную копию всех данных.

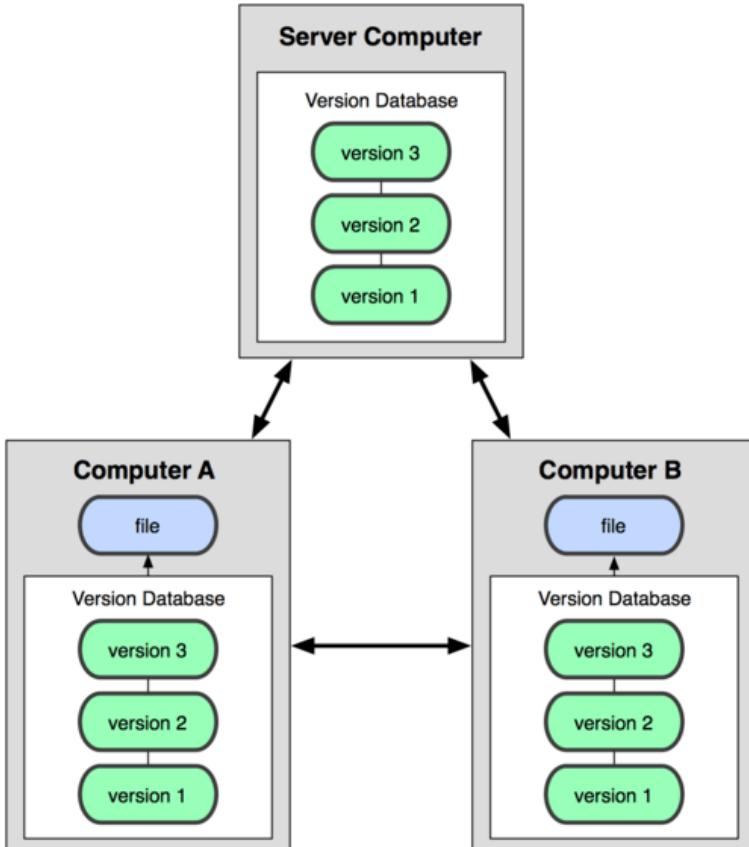


Рисунок 23

В большей части этих систем можно работать с несколькими удаленными репозиториями, таким образом, можно одновременно работать по-разному с разными группами людей в рамках одного проекта. Так, в одном

проекте можно одновременно вести несколько типов рабочих процессов, что невозможно в централизованных системах.

Основы работы с Git

Одним из самых популярных представителей РСКВ является **система Git**.

Целью Git стало создание более быстрой системы контроля версий относительно CVS. Git работает быстрее всего под Linux.

Преимущества Git:

- высокое быстродействие;
- дешевые операции с ветками кода;
- ведется полная история разработки, доступная оффлайн.

Недостатки Git:

- не все функции работают под Windows в сравнении с Linux.

1. Для начала работы с Git нужно сделать его установку:

- под **Linux** – нужно открыть терминал и установить приложение при помощи пакетного менеджера вашего дистрибутива;
- под **Windows** – рекомендуется использовать **git for windows**, он содержит клиент с графическим интерфейсом и эмулятор bash.
- под **OS X** – проще всего воспользоваться **homebrew**.

2. Следующий шаг – настройка.

Есть довольно много опций, но нужно настроить самые важные: имя пользователя и адрес электронной почты. Это позволит регистрировать для каждого действия пользователя отметки его имени и почты, т. е. пользователи всегда будут знать, кто внес какие изменения.

3. Создание нового репозитория.

Если вы собираетесь начать использовать Git для существующего проекта, то вам необходимо перейти в проектный каталог и в командной строке ввести:

```
$ git init
```

4. Определение состояния репозитория.

Команда `status` – это команда, которая показывает информацию о текущем состоянии репозитория: актуальна ли информация на нем, что появилось нового, что изменилось.

Жизненный цикл файлов

Каждый файл в вашем рабочем каталоге может находиться в одном из двух состояний: под версионным контролем (отслеживаемые) и без него (неотслеживаемые). **Отслеживаемые файлы** – это те файлы, которые были в последнем слепке состояния проекта (`snapshot`); они могут быть неизмененными, измененными или подготовленными к коммиту (`staged`).

Неотслеживаемые файлы – это все остальное, любые файлы в вашем рабочем каталоге, которые не входили в ваш последний слепок состояния и не подготовлены к коммиту.

Когда вы впервые клонируете репозиторий, все файлы будут отслеживаемыми и неизмененными, потому что вы только взяли их из хранилища (**checked them out**) и ничего пока не редактировали.

Как только вы отредактируете файлы, Git будет рассматривать их как измененные, т. к. вы изменили их с момента последнего коммита. Вы индексируете (**stage**) эти изменения и затем фиксируете все индексированные изменения, а затем цикл повторяется. Этот жизненный цикл изображен на рисунке 24.

File Status Lifecycle

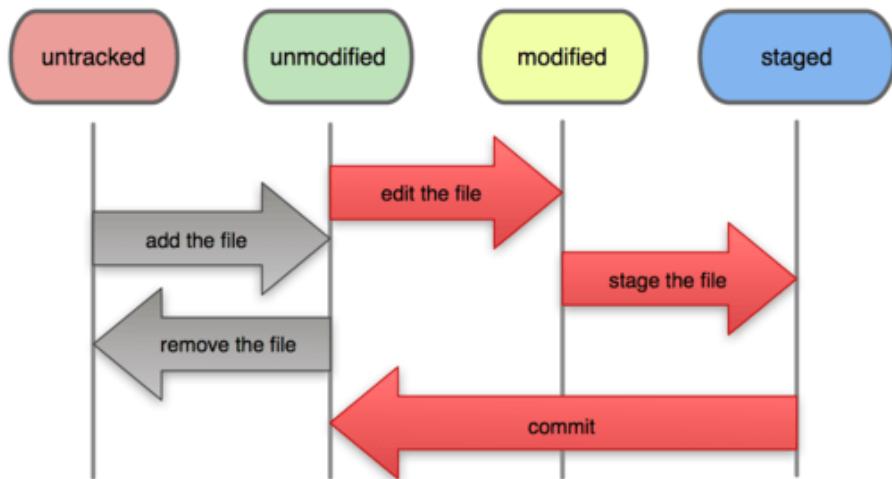


Рисунок 24

1. Определение состояния файлов.

Для определения состояний файлов используется команда **git status**.

2. Отслеживание новых файлов.

Для того чтобы добавить под версионный контроль новый файл, используется команда **git add**. Чтобы начать отслеживание файла **Tutorial**, следует выполнить команду:

```
$ git add Tutorial
```

3. Подготовка файлов.

В Git реализована концепция области подготовленных файлов. Можно представить ее как холст, на который наносят изменения, которые нужны в коммите. Сначала холст пустой, но затем мы добавляем на него файлы (или части файлов, или даже одиночные строчки) командой **add** и, наконец, коммитим все нужное в репозиторий (создаем слепок нужного нам состояния) командой **commit**.

4. Коммит (фиксация изменений).

Коммит представляет собой состояние репозитория в определенный момент времени. Считается хорошей практикой делать коммиты часто и всегда писать содержательные комментарии.

Простейший способ зафиксировать изменения – это набрать команду **git commit**:

```
$ git commit
```

Можно набрать свой комментарий к коммиту в командной строке вместе с командой **commit**, указав его после параметра **-m**, как в следующем примере:

```
$ git commit -m "Story 182: Fix benchmarks for speed"
[master]: created 463dc4f: "Fix benchmarks for speed"
2 files changed, 3 insertions(+), 0 deletions(-)
create mode 100644 README
```

Коммит вывел информацию о себе: на какую ветку выполнен коммит (**master**), какая контрольная сумма SHA-1 у этого коммита (**463dc4f**), сколько файлов было изменено, а также статистику по добавленным/удаленным строкам в этом коммите.

5. Удаление файлов.

Для того чтобы удалить файл из Git'a, необходимо удалить его из отслеживаемых файлов, а затем выполнить коммит. Это позволяет сделать команда **git rm**, которая также удаляет файл из рабочего каталога, так что в следующий раз он же не будет отмечен как «неотслеживаемый».

6. Перемещение файлов.

Если вам хочется переименовать файл в Git'e, можно выполнить команду:

```
$ git mv file_from file_to
```

7. Ветвление.

Во время разработки новой функциональности считается хорошей практикой работать с копией оригинального проекта, которую называют веткой. Ветви имеют свою собственную историю и изолированные друг от друга изменения до тех пор, пока вы не решаете слить изменения вместе. Это происходит по набору причин:

- уже рабочая, стабильная версия кода сохраняется;
- различные новые функции могут разрабатываться параллельно разными программистами;
- разработчики могут работать с собственными ветками без риска, что кодовая база поменяется из-за чужих изменений;

- в случае сомнений, различные реализации одной и той же идеи могут быть разработаны в разных ветках и затем сравниваться.

Участникам команды необходимо научиться делать следующие вещи:

- создание новой ветки;
- переключение между ветками;
- слияние веток.

8. Дополнительные знания, которые могут оказаться полезными.

При совместной работе могут быть полезными такие действия, как:

- отслеживание изменений, сделанные в коммитах;
- возвращение файла к предыдущему состоянию;
- исправление коммита;
- разрешение конфликтов при слиянии.

6. БАГ-ТРЕКЕРЫ

Системы класса «баг-трекер» изначально были придуманы для автоматизации работ по регистрации обнаруженных ошибок и исправлению этих ошибок при разработке программных продуктов.

Продукт **Jira** впервые вышел на рынок в 2003 году и изначально задумывался как система для отслеживания и устранения ошибок при разработке программных продуктов. За десять с небольшим лет Jira превратился в систему, которая может помочь команде планировать и контролировать все виды деятельности при разработке программного обеспечения.

На сегодняшний день Jira входит в «Магический квадрант Gartner» в классе Enterprise Agile Planning Tools, что говорит о его популярности.

Система Jira является облачным сервисом, в основную функциональность, которого входят такие возможности, как:

- использование диаграммы Ганта;
- использование приоритетов для задач;
- использование облачного хранилища;
- уведомления о событиях проекта;
- создание комментариев к задачам;
- вложения файлов к задачам;
- фильтры;
- настройка доступа;

- отслеживание прогресса в процентах;
- отслеживание состояния задач;
- дэшборды;
- управление назначениями;
- база знаний;
- email-уведомления;
- частые вопросы;
- отслеживание потраченного времени;
- разнообразные отчеты.

Для управления Agile-проектом реализована следующая функциональность:

- создание журнала спринта;
- управление бэклогом;
- диаграмма сгорания задач;
- безопасность и конфиденциальность;
- скрам-доска;
- канбан-доска.

С точки зрения поддержки традиционного подхода к управлению проектами в Jira нет следующей функциональности:

- функциональности, связанной с бюджетированием проекта;
- календарей отпусков и выходных дней;
- возможностей создавать диаграмму Ганта.

Часть из этой функциональности можно найти среди платных **addons**, которые расширяют возможности продукта.

Цена на Jira очень привлекательная для команд не более 10-ти человек (\$120 в год) и сильно вырастает для 11-ти и более лицензий (\$7 в месяц на одного пользователя). А если вы хотите использовать сервис для создания базы знаний по проектам – **Confluence**, то будьте готовы выложить еще \$5 в месяц за каждого пользователя. Недостающую функциональность можно купить и интегрировать с помощью **addons**. При этом, бесплатных лицензий у Jira нет.

Основные приемы работы в Jira

Для создания проекта в Jira нужно нажать кнопку **Создать проект**:

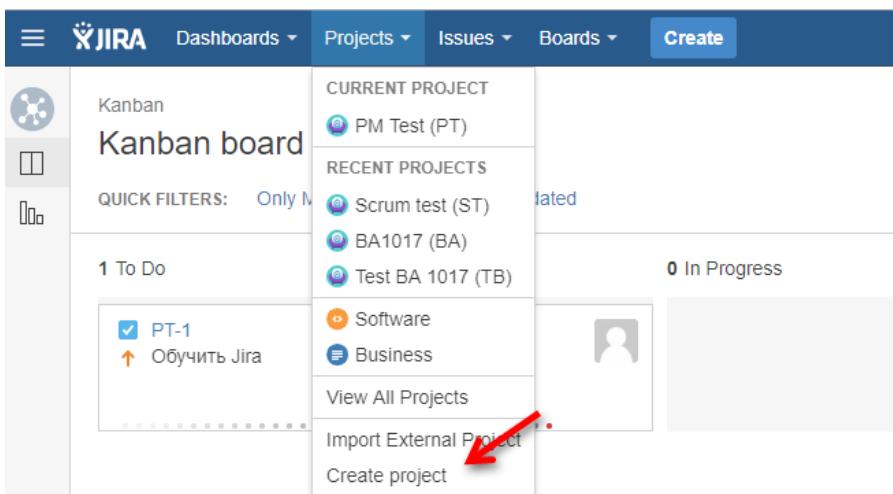


Рисунок 25

Затем нужно выбрать вариант типа проекта: **Scrum**, **Kanban** или, к примеру **Управление проектом**» (см. рис. 26).

Create project

[View Marketplace Workflow](#)

SOFTWARE



Scrum software development

Agile development with a board, sprints and stories. Connects with source and build tools.



Kanban software development

Optimise development flow with a board. Connects with source and build tools.



Basic software development

Track development tasks and bugs. Connects with source and build tools.

BUSINESS



Project management

Plan, track and report on all of your work within a project.



Task management

Quickly organize and assign simple tasks for you and your team.

Рисунок 26

После этого нужно настроить схему статусов в проекте:

Process management

Create your tasks and track them at every step, from start to finish. You can use this project to review documentation, approve expenses, or other processes.

ISSUE TYPES

- Task
- Sub-task

WORKFLOW

[Back](#)[Select](#)[Cancel](#)

Рисунок 27

И ввести параметры проекта:

Process management

Name



Max. 80 characters.

Key



Max. 10 characters.

Project Lead

 admin

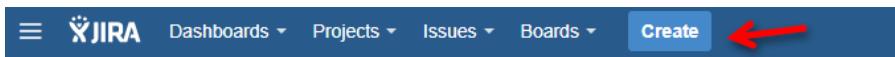

Enter the username of the Project Lead.

Рисунок 28

Если на этапе выбора типа проекта был выбран Scrum, то для такого типа проекта станет доступна Scrum-доска, возможность использовать такие типы активностей как: **Epic, User-story и Task.**

Если был выбран тип проекта Kanban, то можно будет настроить этапы процесса, установить WIP-лимиты для каждого этапа и т. д.

Активность в проекте можно создавать нажатием кнопки **Create:**



PM

Рисунок 29

После чего можно выбрать тип активности и внести данные в поля формы для активности.

Для проектов по Scrum на доске можно использовать механизм перетягивания карточек задач из одной области доски в другую:

A screenshot of a Scrum board in Jira. The board is divided into three columns: 'To Do', 'In Progress', and 'Done'. Each column contains several task cards, each with a checkbox, a title, a brief description, and a 'ServoBox' button. The 'In Progress' column has three cards: EMB 866 (checkbox checked, description: 'Провести испытания сервобокса'), EMB-879 (checkbox checked, description: 'Составить полный список комплекта поставки для шести моторов (проверить на сайте и сделать чек-лист для внутренних нужд)'), and EMB 880 (checkbox checked, description: 'Создать подробный тайориал с фото и видеоролик'). The 'Done' column has three cards: EMB 865 (checkbox checked, description: 'Снять Сервобокс'), EMB-878 (checkbox checked, description: 'B Table 1. Line segment lengths vs. cross-sections исправить для столбца 13: поставить прочерки и добавить нужные размеры'), and EMB 881 (checkbox checked, description: 'проверить чертежи, инструкцию, список компонентов и в PDM-системе присваивать статус "приятно"').

To Do	In Progress	Done
	<p>EMB 866 Провести испытания сервобокса ServoBox</p> <p>EMB-879 Составить полный список комплекта поставки для шести моторов (проверить на сайте и сделать чек-лист для внутренних нужд) ServoBox</p> <p>EMB 880 Создать подробный тайориал с фото и видеоролик ServoBox</p>	<p>EMB 865 Снять Сервобокс ServoBox</p> <p>EMB-878 B Table 1. Line segment lengths vs. cross-sections исправить для столбца 13: поставить прочерки и добавить нужные размеры ServoBox</p> <p>EMB 881 проверить чертежи, инструкцию, список компонентов и в PDM-системе присваивать статус "приятно" ServoBox</p>

Рисунок 30

По каждой задаче можно вести учет фактически потраченного и оставшегося времени, при этом можно будет использовать диаграмму BurnDown для контроля спринта:

EMB Sprint 18 ▾

Закрытый спринт, завершил: Max 11/июн/18 1:25 PM - 19/июн/18 1:33 PM связанных страниц

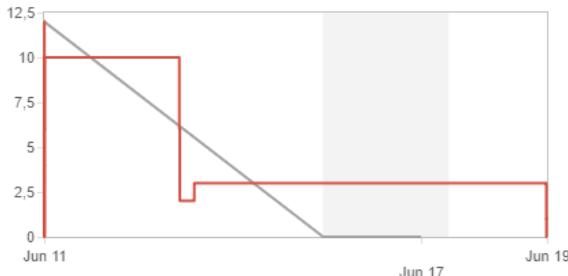


Рисунок 31

Для завершения спринта нужно нажать кнопку:



Рисунок 32

Для того чтобы овладеть основными приемами работы по фреймворку Scrum пользователю, на наш взгляд, нужно пару часов активной работы в Jira, и все станет понятно.

Следует отметить, что настройка и администрирование Jira является достаточно сложным процессом, и чтобы разобраться в них, могут потребоваться сотни часов.

Mantis BT

Mantis BT – бесплатный bugtracker, который позволяет создавать сообщения об ошибках и отслеживать дальнейший процесс работы с ними; можно использовать как систему учета обращений и инцидентов. Продукт можно интегрировать с wiki-движком для создания документации (**DokuWiki**).

Система Mantis BT является облачной и работает через веб-браузер. К плюсам можно отнести следующие:

- продукт бесплатный;
- код на PHP свободно модифицируем;
- настраиваемые пользователем поля;
- удобные фильтры.

7. СИСТЕМЫ УПРАВЛЕНИЯ ПРОЕКТАМИ

MS Project

MS Project – продукт компании Microsoft долгое время считался одним из самых функциональных продуктов для управления сроками и бюджетом проектов. На рынке однопользовательских решений Microsoft Project до 2010 года был лидером, зарабатывая на продажах более 900 млн долларов в год и имея клиентскую базу в 20 000 000 пользователей.

Функциональность программы поддерживает практически все методы и инструменты управления сроками и бюджетом проектов, описанные в PMBOK, такие как:

Планирование проекта

Декомпозиция задач

Оценки трудоемкости задач

Сетевой график и метод критического пути

Структуры ресурсов

Функциональность для оптимизации ресурсов для проекта

Выравнивание загрузки ресурсов

Планирование затрат на проект

Оценка влияния рисков через имитационное моделирование «а что если?»

Интерактивная оптимизация планов проекта

Отслеживание и контроль проекта

Возможность сохранения базового плана проекта

План/фактный анализ

Возможность ввода фактических и оставшихся тру-
дозатрат по задачам проекта

Диаграмма Ганта с отслеживанием

Поддержка Метода освоенного объема

Отметим, что для моделирования проекта с учетом ограниченных ресурсов MS Project является одним из самых функциональных программных продуктов в мире. Однако для коллективной работы над проектом версии **MS Project Standart** – недостаточно. Поэтому разработчики продукта предлагают **MS Project Server**, который предполагает клиент-серверную архитектуру, или облачный продукт **Project Online**.

Основной выгодой от использования MS Project является возможность создать модель проекта, учитывающую имеющиеся ограничения. Рассмотрим пример создания такой модели проекта.

Пример использования MS Project для планирования ИТ-проекта

Изучать возможности MS Project мы будем на примере версии **MS Project 2013 Standart**.

Допустим, в ИТ-компанию поступил запрос от клиента, который планирует реализовать проект внедрения ИТ-системы за 3 месяца. Руководителю ИТ-компании необходимо понять, реален ли этот срок, с учетом загрузки сотрудников на других проектах.

- Для ответа на этот вопрос руководитель проекта сделал модель проекта в MS Project.

При входе в программу по умолчанию открывается представление **Диаграмма Ганта**:

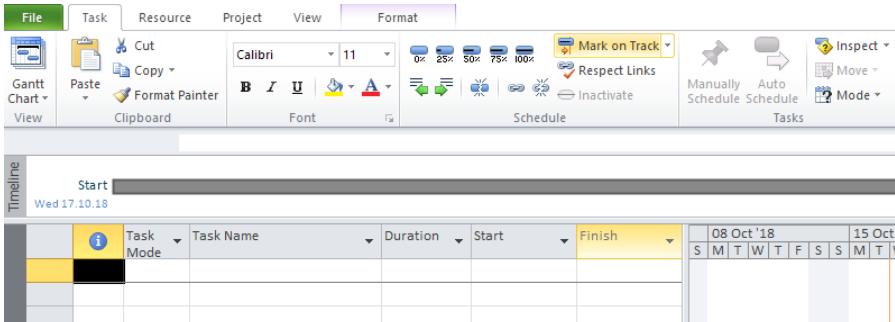


Рисунок 33

Переключение между представлениями – это таблица или диаграмма, которую пользователь видит на экране – в MS Project осуществляется нажатием кнопки **Диаграмма Ганта**.

Сначала руководитель проекта в представлении **Диаграмма Ганта** разработал структуру работ по проекту, используя возможности создания иерархии задач в проекте. Для этого в таблицу **Диаграмма Ганта**, он ввел список задач по проекту и, использовал кнопку **Понизить уровень** для создания иерархии в списке:

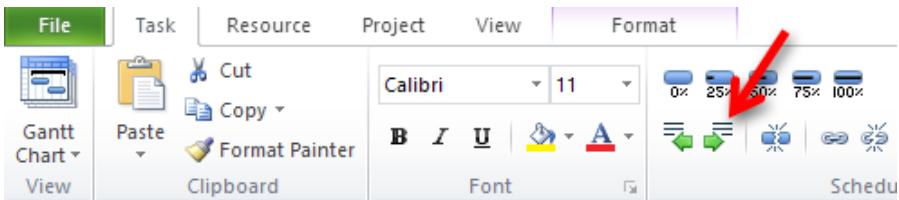


Рисунок 34

В итоге получилась следующая иерархия работ по проекту:

Управление проектом	0 дней?	Пт 14.09.18	Пт 14.09.18
Планирование спринтов, ретроспективы, стендапы	0 дней?	Пт 14.09.18	Пт 14.09.18
Встречи с заказчиком по статусу проекта	0 дней?	Пт 14.09.18	Пт 14.09.18
Разработка и согласование актов	0 дней?	Пт 14.09.18	Пт 14.09.18
□ Доработки и внедрение модуля Зарплата		Пт 14.09.18	
□ Первоначальное заполнение информационной базы		Пт 14.09.18	
Анализ требований и написание ТЗ	0 дней?	Пт 14.09.18	Пт 14.09.18
Загрузка	0 дней?	Пт 14.09.18	Пт 14.09.18
Тестирование	0 дней?	Чт 20.09.18	Чт 20.09.18
Исправление ошибок	0 дней?	Чт 20.09.18	Чт 20.09.18
Первоначальная настройка системы по учетным политикам 3-ех компаний	0 дней?	Пт 14.09.18	Пт 14.09.18
□ Оклады в иностранной валюте		Пт 14.09.18	
Анализ требований	0 дней?	Пт 14.09.18	Пт 14.09.18
Доработка системы расчетов	0 дней?	Пт 14.09.18	Пт 14.09.18
Тестирование	1 день?	Пт 14.09.18	Пт 14.09.18
Демо	1 день?	Пн 17.09.18	Пн 17.09.18
Исправление ошибок	1 день?	Вт 18.09.18	Вт 18.09.18

Рисунок 35

После этого, в таблицу было добавлено поле **Трудозатраты** (щелкаем правой кнопкой мыши на названии поля **Длительность**, выбираем **Добавить столбец** и в выпадающем списке выбираем поле **Трудозатраты**), и для каждой задачи руководитель проекта внес данные о плановых трудозатратах на задачу в человеко-часах (см. рис. 36).

7. Системы управления проектами

Name	Work	Duration	Start	Finish
+ Управление проектом	30 h	50 days	Fri 14.09.18	Thu 22.11.18
□ Доработки и внедрение модуля Зарплата	205 h	56 days?	Fri 14.09.18	Fri 30.11.18
□ Первоначальное заполнение информационной базы	70 h	35 days	Mon 17.09.18	Fri 02.11.18
Анализ требований и написание ТЗ	10 h	5 days	Mon 17.09.18	Fri 21.09.18
Загрузка	40 h	25 days	Mon 24.09.18	Fri 26.10.18
Тестирование	10 h	20 days	Mon 01.10.18	Fri 26.10.18
Исправление ошибок	10 h	5 days	Mon 29.10.18	Fri 02.11.18
Первоначальная настройка системы по учетным политикам 3-ех компаний	25 h	20 days	Mon 24.09.18	Fri 19.10.18
□ Оклады в иностранной валюте	10 h	13 days	Fri 14.09.18	Tue 02.10.18
Анализ требований	2 h	5 days	Fri 14.09.18	Thu 20.09.18
Доработка системы расчетов	4 h	2 days	Fri 21.09.18	Mon 24.09.18
Тестирование	1 h	2 days	Tue 25.09.18	Wed 26.09.18
Демо	1 h	2 days	Thu 27.09.18	Fri 28.09.18
Исправление ошибок	2 h	2 days	Mon 01.10.18	Tue 02.10.18
□ Доработка учета доп.привилегий (благ) сотрудникам, компенсационные выплаты	15 h	15 days	Wed 03.10.18	Tue 23.10.18
Анализ требований	2 h	4 days	Wed 03.10.18	Mon 08.10.18

Рисунок 36

Вы можете заметить, что поле **Длительность**, при такой очередности создания модели проекта, в каждой задаче имеет значение **1 день?**. Наличие знака **?** в этом поле говорит о том, что длительность выполнения задачи создана автоматически и пользователь данные в этом поле не менял.

Расчет длительности задач выполним автоматически после определения трудозатрат на задачи и назначения исполнителей задач.

2. На следующем шаге руководитель проекта создал список участников проекта в представлении **Лист ресурсов**.

Для этого, он нажал на кнопку **Диаграмма Ганта** и выбрал из списка выпадающего списка команду **Лист ресурсов**:

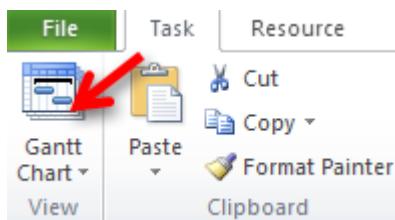


Рисунок 37

При этом он внес по каждому из них стоимость чел.-часа (поле **Стандартная ставка**) и % доступного времени, на который участник выделен на проект (поле **Макс. единиц**):

Name	Type	Max Units	Standard Rate
Разработчик 1	Work	30%	\$13,00/h
Разработчик 2	Work	50%	\$13,00/h
Аналитик	Work	70%	\$5,00/h
Руководитель проекта	Work	20%	\$30,00/h

Рисунок 38

Данные в поле **Макс. единиц** обозначают, какую долю от внесенного в календарь сотрудника доступного рабочего времени в день, он может работать в данном проекте.

Например, Разработчик 1 при 8-часовом рабочем дне может тратить на задачи проекта не более 2,4 часа в день. Если на него назначить задач на большее время в день,

то программа определит, что данный ресурс перегружен и выдаст об этом сообщение.

- После этого, руководитель проекта вернулся в представление **Диаграмма Ганта** и назначил участников проекта на задачи проекта.

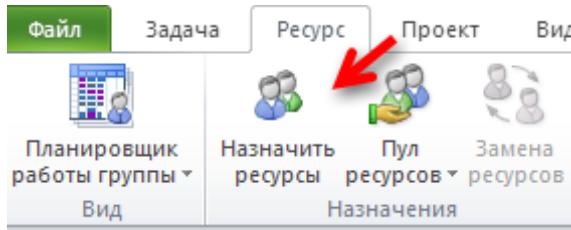


Рисунок 39

При назначении исполнителя программа сама рассчитывает длительность задач, учитывая их трудозатраты и процент загрузки исполнителя, указанный в поле **Макс. единиц** в представлении **Лист ресурсов**.

В результате руководитель проекта получил следующие расчеты по длительности задач проекта:

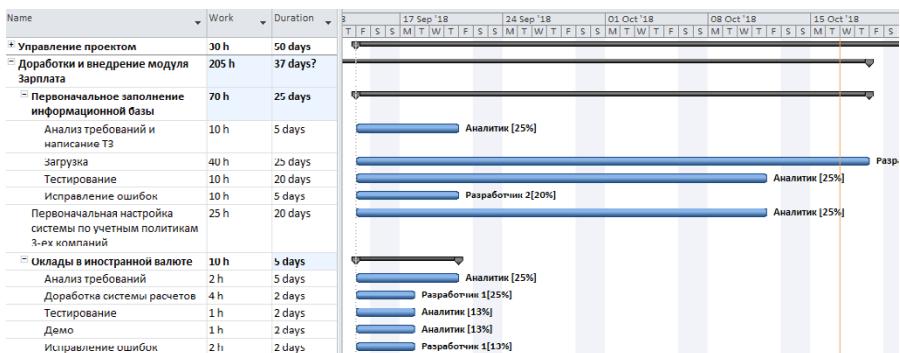


Рисунок 40

На диаграмме справа от таблицы (см. рис. 40) видна длительность каждой задачи на отрезке временной шкалы, из диаграммы можно узнать дату старта всех задач проекта и их финиш – это диаграмма Ганта.

4. На следующем шаге руководитель проекта внес связи между задачами проекта.

Для этого выделяется задача, которая является предшественником и через клавишу **Ctrl** выделяется вторая задача, которая является последователем, после этого нажимаем кнопку **Связать задачи**:

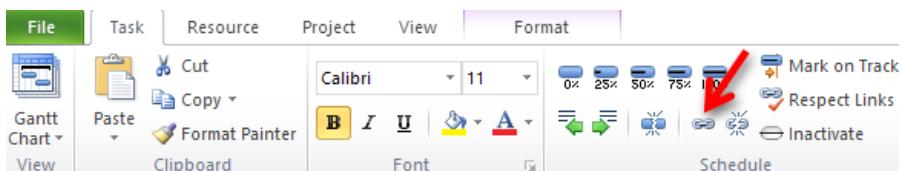


Рисунок 41

И получил следующий прогнозный срок реализации проекта (см. рис. 42).

Как видно из рисунка, на диаграмме Ганта появились связи между задачами и, если эти связи определены верно, то руководитель проекта может предположить, что самой оптимистичной датой финиша для этого проекта является 29.11.2018 года.

5. Для проверки корректности модели проекта нужно убедиться в том, что ни один из участников проекта не имеет перегрузок.

В программе есть такая возможность при включении вида **Использование ресурсов**. Как видно из таблицы (см. рис. 43), ни один ресурс не отмечен красным цветом – это обозначает, что ни у кого из ресурсов нет перегрузки.

Но и этот факт не делает модель проекта устойчивой к воздействиям внешней и внутренней среды, поэтому для повышения вероятности успеха проекта нужно спланировать риски, внести антирисковые мероприятия в модель проекта и пересчитать плановые сроки и бюджет.

Name	Work	Duration	Start	Finish
Аutomатизация учета	235 h	56 days?	Fri 14.09.18	Fri 30.11.18
+ Управление проектом	30 h	50 days	Fri 14.09.18	Thu 22.11.18
Доработки и внедрение модуля Зарплата	205 h	56 days?	Fri 14.09.18	Fri 30.11.18
Первоначальное заполнение информационной базы	70 h	35 days	Mon 17.09.18	Fri 02.11.18
Анализ требований и написание ТЗ	10 h	5 days	Mon 17.09.18	Fri 21.09.18
Загрузка	40 h	25 days	Mon 24.09.18	Fri 26.10.18
Тестирование	10 h	20 days	Mon 01.10.18	Fri 26.10.18
Исправление ошибок	10 h	5 days	Mon 29.10.18	Fri 02.11.18
Первоначальная настройка системы по учетным политикам 3-ех компаний	25 h	20 days	Mon 24.09.18	Fri 19.10.18
Оклады в иностранной валюте	10 h	13 days	Fri 14.09.18	Tue 02.10.18
Анализ требований	2 h	5 days	Fri 14.09.18	Thu 20.09.18
Доработка системы расчетов	4 h	2 days	Fri 21.09.18	Mon 24.09.18
Тестирование	1 h	2 days	Tue 25.09.18	Wed 26.09.18
Демо	1 h	2 days	Thu 27.09.18	Fri 28.09.18
Исправление ошибок	2 h	2 days	Mon 01.10.18	Tue 02.10.18

Рисунок 42

Название ресурса	Work
+ Unassigned	0 h
+ Разработчик 1	31 h
+ Разработчик 2	88 h
+ Аналитик	86 h
+ Руководитель проекта	30 h

Рисунок 43

6. Для расчета себестоимости проекта руководитель проекта добавляет в таблицу поле **Затраты и получает данные о внутренней стоимости проекта:**

Name	Work	Duration	Cost
☐ Автоматизация учета	235 h	56 days?	\$2 877,00
+ Управление проектом	30 h	50 days	\$900,00
☒ Доработки и внедрение модуля Зарплата	205 h	56 days?	\$1 977,00
☒ Первоначальное заполнение информационной базы	70 h	35 days	\$750,00
Анализ требований и написание ТЗ	10 h	5 days	\$50,00
Загрузка	40 h	25 days	\$520,00
Тестирование	10 h	20 days	\$50,00
Исправление ошибок	10 h	5 days	\$130,00
Первоначальная настройка системы по учетным политикам 3-ех компаний	25 h	20 days	\$125,00
☒ Оклады в иностранной валюте	10 h	13 days	\$98,00
Анализ требований	2 h	5 days	\$10,00
Доработка системы расчетов	4 h	2 days	\$52,00
Тестирование	1 h	2 days	\$5,00
Демо	1 h	2 days	\$5,00
Исправление ошибок	2 h	2 days	\$26,00

Рисунок 44

Описанный кейс демонстрирует основные возможности MS Project как программы для планирования и оценки сроков и ресурсов ИТ-проекта, которая позволяет осуществлять моделирование сценариев «А что будет со сроками, если на проект выделить такое количество людей?».

Также программа позволяет рассчитать бюджет проекта при наличии оценок трудоемкости по задачам проекта и стоимости человека-часа у участников проекта. При этом в программе есть возможность ввести фиксированную стоимость затрат на задачу проекта или этап (например, стоимость контракта по выполнению одной из задач).

После того как рассчитанные сроки и бюджет проекта устраивают заказчика проекта, руководитель проекта сохраняет базовый план проекта и имеет возможность проводить план-фактный анализ по срокам и бюджету. Например, для анализа хода проекта по срокам можно использовать **Диаграмму Ганта с отслеживанием**:

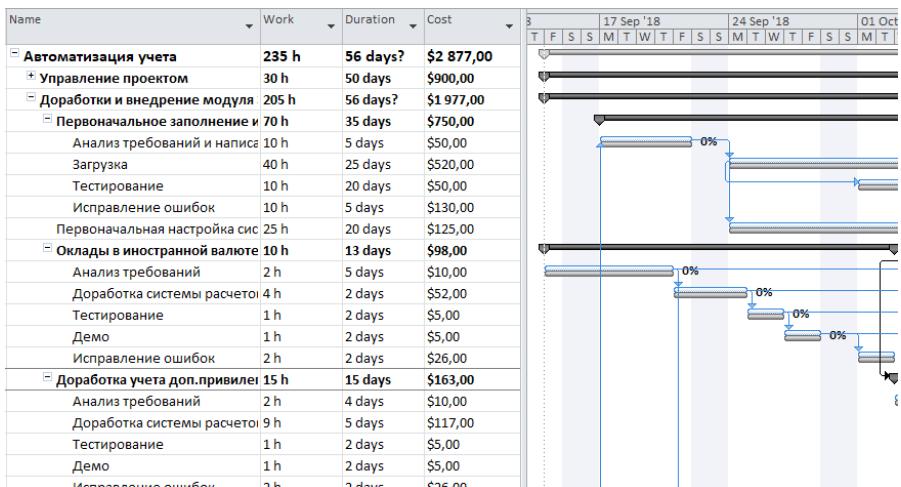


Рисунок 45

На диаграмме серым цветом отражены временные отрезки для базового плана, а синим и красным – отрезки для текущего плана. Из диаграммы видно, что проект уже отстает от базового плана по срокам.

В таблице **Затраты** можно получать данные об отклонении базовых затрат по задачам проекта от фактических:

Task Name	Fixed Cost	Fixed Cost Accrual	Total Cost	Baseline	Variance	Actual
+ Управление проектом	\$0,00	Prorated	\$900,00	\$900,00	\$0,00	\$0,00
Доработки и внедрение модуля	\$0,00	Prorated	\$1 977,00	\$1 977,00	\$0,00	\$0,00
Первоначальное заполнение	\$0,00	Prorated	\$750,00	\$750,00	\$0,00	\$0,00
Анализ требований и	\$0,00	Prorated	\$50,00	\$50,00	\$0,00	\$0,00
Загрузка	\$0,00	Prorated	\$520,00	\$520,00	\$0,00	\$0,00
Тестирование	\$0,00	Prorated	\$50,00	\$50,00	\$0,00	\$0,00
Исправление ошибок	\$0,00	Prorated	\$130,00	\$130,00	\$0,00	\$0,00
Первоначальная настройка системы по	\$0,00	Prorated	\$125,00	\$125,00	\$0,00	\$0,00
Оклады в иностранной валюте	\$0,00	Prorated	\$98,00	\$98,00	\$0,00	\$0,00
Анализ требований	\$0,00	Prorated	\$10,00	\$10,00	\$0,00	\$0,00
Доработка системы	\$0,00	Prorated	\$52,00	\$52,00	\$0,00	\$0,00
Тестирование	\$0,00	Prorated	\$5,00	\$5,00	\$0,00	\$0,00
Демо	\$0,00	Prorated	\$5,00	\$5,00	\$0,00	\$0,00
Исправление ошибок	\$0,00	Prorated	\$26,00	\$26,00	\$0,00	\$0,00

Рисунок 46

Долгие годы MS Project был одним из самых популярных в своем классе в мире.

Однако для коллективной работы MS Project требовал установки и администрирования MS Project Server, что само по себе было довольно сложным для ИТ-администраторов.

В 2006-2007 годах на рынке систем управления проектами стали появляться облачные продукты для управления проектами, которые к началу 2010-х годов стали вытеснять системы, требовавшие установки на сервера компаний. Разработчики MS Project пропустили этот тренд и к моменту выхода MS Project Online – облачного MS Project – сражение уже было проиграно. В последние

5 лет, начиная с 2013 года, Microsoft Project Online не входит в Magic Quadrant for Cloud-Based IT Project and Portfolio Management Services компании Gartner.

Поэтому многие ИТ-компании используют MS Project на этапе планирования проекта, а для коллективной работы используют облачные продукты, такие как **Jira** или **Easy Projects**. В Easy Projects есть возможность выгрузить файл, созданный в MS Project и дальше уже работать совместно с командой над задачами этого проекта, внося изменения в общий план проекта.

Easy Projects

Easy Projects – облачный продукт, поддерживающий как традиционные инструменты управления проектами (диаграмма Ганта, методика освоенного объема и т. д.), так и Agile-подходы, например Kanban.

Основными преимуществами данного сервиса является простота в использовании, сбалансированная функциональность и наличие как платных, так и бесплатных рабочих мест, чего нет у большинства конкурентов.

В Easy Projects есть интеграция с MS Project, благодаря которой можно загрузить файл, созданный в MS Project, на портал Easy Projects, и дальше работать с проектом уже в среде Easy Projects.

Разработано большое количество интеграций с популярными продуктами, такими как мессенджер **Slack**, **MS Outlook**, **Power BI** (одна из лучших BI систем в мире) и сервис **Zappier**, который позволяет самостоятельно создавать интеграцию со многими сервисами, не требуя для этого навыков программирования.

Среди клиентов Easy Projects есть такие известные компании, как: Lenovo, Hewlett Packard, Symantec, Toshiba и другие.

Стоимость Easy Projects начинается с \$10 в месяц на одного пользователя и до \$26 за пользователя, в зависимости от функционала. При этом есть бесплатные лицензии, позволяющие их пользователям видеть список задач, назначенных на них, прикладывать файлы к задачам, завершать задачи и комментировать их.

Team Foundation Server

Team Foundation Server (TFS) – продукт корпорации Microsoft, представляющий собой комплексное решение, объединяющее в себе систему управления версиями, сбор данных, построение отчетов, отслеживание статусов и изменений по проекту и предназначено для совместной работы над проектами по разработке программного обеспечения.

TFS позволяет автоматизировать следующие функции:

- **контроль исходного кода (Source control).**
- **отчетность (Reporting).**

При помощи слоя **Отчетности** можно создавать множество отчетов на основе объединения информации о рабочих элементах, наборах изменений, информации. Отчеты, созданные при помощи **SQL Server Reporting Services**, можно экспортить в нескольких различных форматах, включая Excel, XML, PDF и TIFF. Отчеты можно просматривать как при помощи Visual Studio, так и через веб-портал.

- **портал проекта (Project portal).**

TFS использует возможности **SharePoint** для создания сайта для проекта, который может использоваться для отслеживания прогресса проекта, наблюдения за рабочими элементами и документами, представленными в библиотеке проекта.

- **Shared services.**

TFS обеспечивает поддержку множества сервисов, которые могут быть использованы для интеграции с посторонними приложениями, как, например, системы управления проектами.

Для организации работы команды по Scrum с помощью TFS нужно использовать шаблон процесса Scrum. Как утверждает Microsoft, этот шаблон был разработан с участием экспертов из компании **Scrum.Org** – одной из ведущих консалтинговых компаний в мире, специализирующихся на Scrum.

Шаблон состоит из набора следующих рабочих элементов:

- **Product Backlog Item (PBI):** Отслеживание требований ко всему программному продукту.
- **Bug:** Найденная в процессе разработки ошибка.
- **Sprint:** В этом рабочем элементе фиксируются даты спринта, цели и ретроспектива. В них спринт обозначается только с помощью механизма иерархий итераций, но, к сожалению, с помощью него невозможно обозначить даты начала и окончания спринта.
- **Impediment:** Проблемы, риски и все то, что может влиять на работу команды, но не связано напрямую

со свойствами разрабатываемого продукта.

- **Test Case:** Описание теста для PBI. Обычно тесты напрямую привязываются к конкретному PBI. Этот рабочий элемент (как, впрочем, и Shared Steps) не определен напрямую в Scrum, но он необходим для корректной работы инфраструктуры TFS.
- **Shared Steps:** Разделяемые между несколькими Test Case шаги проверок.

Для планирования спринтов команды используют типы рабочих элементов «Элемент невыполненной работы» (PBI) и «Ошибка» (Bug):

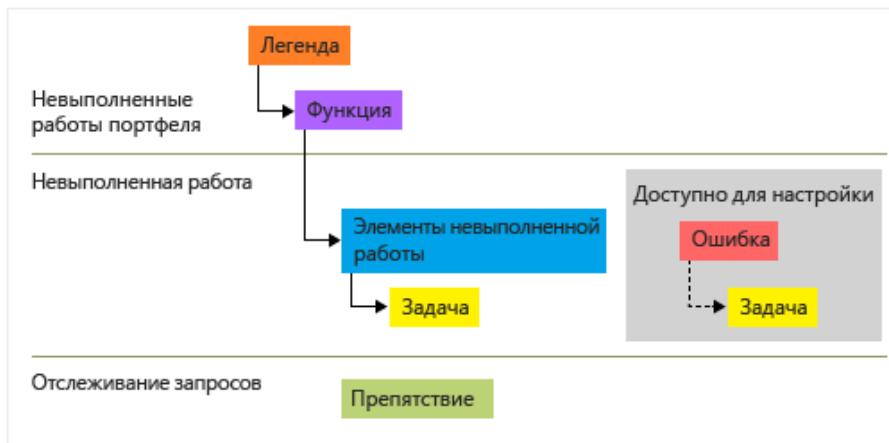


Рисунок 47

Отметим, что **Product Backlog Item** и **Bug** – это равноценные элементы списка журнала продукта с точки зрения оценок затрат и планирования, поэтому они имеют практически идентичный набор полей.

Task: Задача. Этот элемент позволяет внести требуемый уровень детализации для **Product Backlog Item**.

Естественно, и сами задачи могут быть раздроблены на подзадачи.

Элементы **Product Backlog Item** и **Bug** можно создавать на панели быстрого добавления на странице невыполненной работы по продукту:

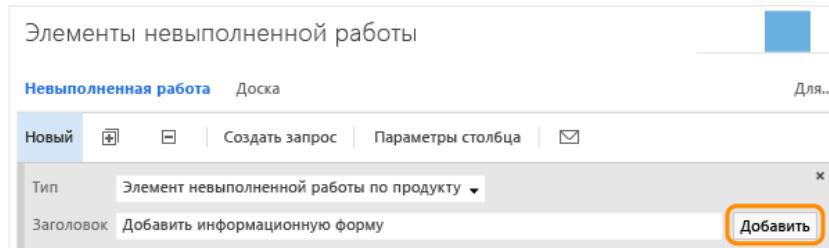


Рисунок 48

Для каждого PBI и ошибки в поле **Приоритет невыполненной работы** владелец продукта может внести данные о приоритете. Кроме этого, на форме **Элемента невыполненной работы** (см. рис. 49) команда разработки заполняет следующие реквизиты:

Поле/вкладка	Использование
Трудозатраты	Оценка объема работ для выполнения PBI. Используются любые единицы измерения: Story points, человеко-часы. Это обязательное поле для заполнения данными отчетов Выполнение выпуска и Скорость
Ценность для бизнеса	Число, которое отражает относительную ценность PBI по сравнению с другими PBIs. Чем больше число, тем выше ценность для бизнеса
Описание (PBI)	Описание истории пользователя или ошибки. Нужно внести столько сведений, чтобы команда могла создавать задачи и тестовые случаи для реализации элемента
Условия приемки	Следует описать критерии, которые команда должна использовать для того, чтобы проверить, полностью ли реализован элемент PBI или исправленная ошибка

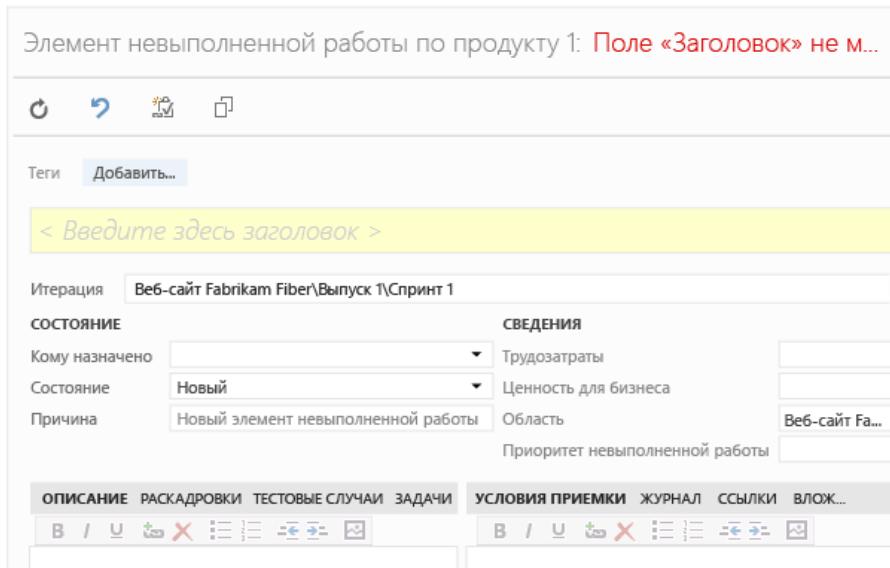


Рисунок 49

Организация работы по Scrum в TFS

1. Создается первичный список **PBI**, по каждому из элементов назначаются приоритеты.
2. Из списка **PBI** отбираются самые важные элементы, делается оценка их сложности (поле **Effort**). Те **PBI**, по которым оценка затруднена (не понятны или нет ответов на часть вопросов), отправляются на дополнительную проработку.
3. Команда формирует первый спринт на основе приоритетов и значимости **PBI** из полного набора **Product Backlog**, и которые уже понятно, как реализовать.
4. **PBI**, которые попали в **Sprint**, делятся на задачи (**Task**). Для задач вводится плановое время реализации.

- Обнаруженные ошибки фиксируются с помощью элемента **Bug**, они тоже попадают в **Product Backlog**.
 - Выполненные **PBI** и ошибки закрываются с статусом **Done**.
 - Планируется следующий Sprint.

Отслеживание хода выполнения спринта

Команды могут использовать канбан-доску для отслеживания хода выполнения PBI и ошибок. При перетаскивании элементов в новый столбец значения в поле **Состояние** обновляются:

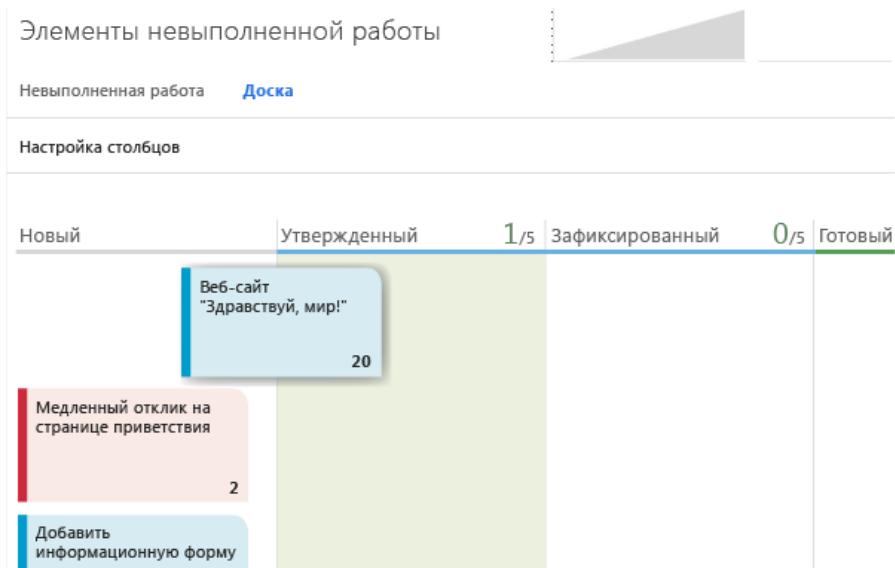


Рисунок 50

Элемент переводится в состояние **Готово**, когда завершены все связанные с ним задачи и владелец продукта принял PBI согласно Условиям приемки.

Отчеты в шаблоне Scrum

Практически все поля, которые определены в рабочих элементах Scrum, попадают в системы аналитики Team Foundation Server и позволяют построить несколько важных отчетов, которые впоследствии дают возможность понимать, как идут дела на проекте.

1. Release Burndown.

От спринта к спринту общий объем работ должен уменьшаться.

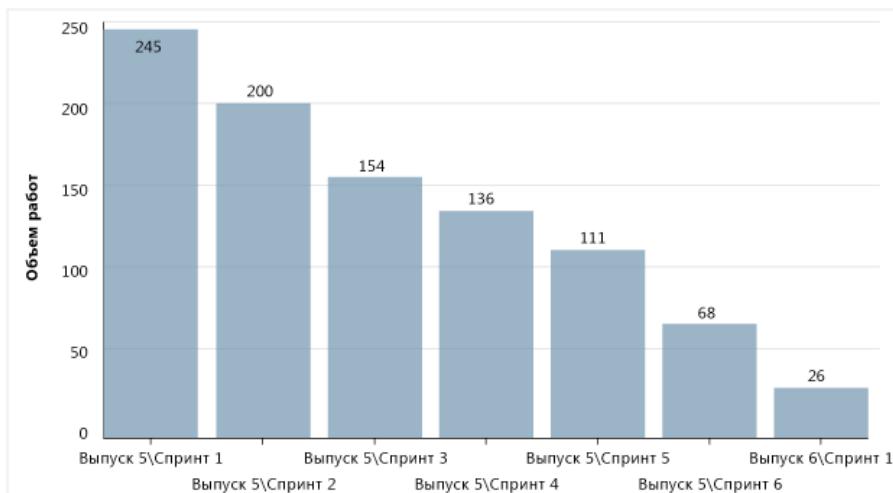


Рисунок 51

2. Отчет Velocity.

Диаграмма скорости показывает объем плановых трудозатрат команды по завершенным в спринте задачам. Исходные данные берутся из журнала невыполненных работ по продукту. Каждый спринт, который был реализован, отображается по горизонтальной оси. На вертикальной оси показывается сумма планового объема работ по задачам,

которые были закрыты (**Состояние = Готово**). Значения по вертикальной оси указываются в тех единицах, которые используются командой (например, человеко-часы, story points).

На диаграмме отображается горизонтальная линия, показывающая среднюю скорость по всем спринтам:

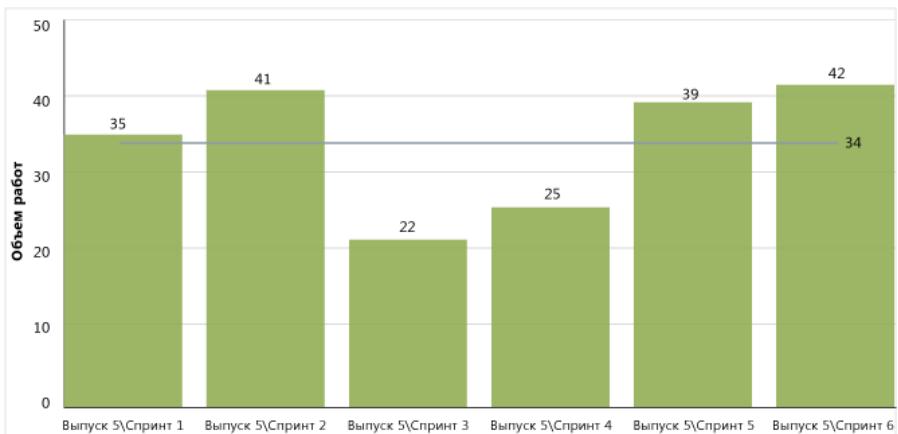


Рисунок 52

3. Build Success Over Time, Build Summary отчеты.

Показывают состояние сборок проекта в разрезе платформ и пройденных тестов. Позволяют понимать прогресс, масштаб изменений и оценивать качество продукта.

Шаблон Scrum, на наш взгляд, имеет достаточно функциональности, чтобы организовать и автоматизировать работу команды по фреймворку Scrum с использованием TFS.

8. ДОМАШНЕЕ ЗАДАНИЕ

Задание 1

Установите себе триальную версию MS Project 2016. Внесите список задач, связи между задачами и длительность задач, как описано в таблице ниже.

Задача-предшественник – задача, которая должна полностью завершиться и по ней должен быть получен результат до старта задачи, для которой определяются связи.

Код задачи	Длительность	Задача-предшественник
A	2	–
B	15	A
C	10	A
D	13	A
E	18	A
F	15	C, D
G	10	F, B
H	5	E, G

Какой получился срок реализации проекта?

Задание 2

Создайте в списке ресурсов ресурс с названием «Программист» и максимальной загрузкой 100%.

Назначьте его на все задачи проекта.

Выровняйте загрузку этого ресурса с помощью кнопки **Выровнять все** на закладке **Ресурсы**.

Получилось ли устраниТЬ перегрузку ресурса?

Какой получился срок реализации проекта после выравнивания?



Урок № 4

ПОДРОБНЕЕ О SCRUM

© Максим Якубович
© Компьютерная Академия «Шаг»
www.itstep.org

Все права на охраняемые авторским правом фото-, аудио- и видеопроизведения, фрагменты которых использованы в материале, принадлежат их законным владельцам. Фрагменты произведений используются в иллюстративных целях в объеме, определенном поставленной задачей, в рамках учебного процесса и в учебных целях, в соответствии со ст. 1274 ч. 4 ГК РФ и ст. 21 и 23 Закона Украины «Про авторське право і суміжні права». Объем и способ цитируемых произведений соответствует принятым нормам, не наносит ущерба нормальному использованию объектов авторского права и не ущемляет законные интересы автора и правообладателей. Цитируемые фрагменты произведений на момент использования не могут быть заменены альтернативными, не охраняемыми авторским правом аналогами, и как таковые соответствуют критериям добросовестного использования и честного использования.

Все права защищены. Полное или частичное копирование материалов запрещено. Согласование использования произведений или их фрагментов производится с авторами и правообладателями. Согласованное использование материалов возможно только при указании источника.

Ответственность за несанкционированное копирование и коммерческое использование материалов определяется действующим законодательством Украины.