Министерство науки и высшего образования Российской Федерации
Федеральное государственное бюджетное образовательное учреждение
высшего образования «Кубанский государственный университет»
(ФГБОУ ВО «КубГУ»)

МЕТОДИЧЕСКИЕ УКАЗАНИЯ

по выполнению практических работ по дисциплине
«Коллективная разработка приложений»

для студентов направлений подготовки 01.03.02 «Прикладная математика и информатика» профиль «Программирование и информационные технологии», 02.03.03 «Математическое обеспечение и администрирование информационных систем» профиль «Технология программирования», 09.03.04 «Программная инженерия» профиль Разработка программного обеспечения и приложений искусственного интеллекта

Краснодар
2025

УДК 629.113(071):004.01:004.4

Методические указания к выполнению практических работ по дисциплине «Коллективная разработка приложений», для студентов направлений подготовки 01.03.02 «Прикладная математика и информатика» профиль «Программирование и информационные технологии», 02.03.03 «Математическое обеспечение и администрирование информационных систем» профиль «Технология программирования», 09.03.04 «Программная инженерия» профиль Разработка программного обеспечения и приложений искусственного интеллекта / Составитель: доц. каф. ИТ Полетайкин А.Н. − Краснодар: КубГУ, 2025. – 25 с.

Методические указания содержат теоретические положения, задания и методические указания к выполнению практических работ по дисциплине «Коллективная разработка приложений».

Составитель: доц. каф. ИТ, к.т.н., доц. Полетайкин А.Н.

СОДЕРЖАНИЕ

ОБЩИЕ ПОЛОЖЕНИЯ

Данный лабораторный практикум рассматривает подходы к организации коллективной разработки программных приложений компьютерных систем. Основное внимание уделяется методологии и решениям Microsoft в части управления жизненным циклом программных приложений: Visual Studio и Azure DevOps Server.

Для выполнения лабораторных работ необходимо ознакомиться с курсом лекций по данной дисциплине, в котором рассматриваются современные технологии разработки программного обеспечения (ПО), процессы коллективной разработки ПО, анализируются формальные и гибкие технологии разработки ПО информационных систем, при обзоре инструментальных средств основное внимание уделяется архитектуре и функциональным возможностям Visual Studio и Azure DevOps Server – ADS (до 2019 года – Team Foundation Server, TFS), организации коллективной разработки на базе Visual Studio, обеспечению качества ПО.

Лабораторный практикум предусматривает выполнение 7 лабораторных работ, которые предполагают коллективную разработку программной системы с использованием Visual Studio и Azure DevOps Server. Непосредственно разработка ПО осуществляется в соответствии с индивидуальным заданием. Рабочая схема выполнения лабораторных работ представлена в табл. 1.

Таблица 1 – Рабочая схема выполнения лабораторных работ

№ л. р.	Тема лабораторной работы: материал для изучения и характер выполняемых работ	Объем, часов	Неделя выполнения
1.	Создание командного проекта: установка и подключение к TFS, конфигурирование TFS, создание командного проекта, настройка области и итераций, настройка параметров команды, настройка оповещений, веб-доступ к параметрам проекта	4	1
2.	Разработка требований к программному приложению: создание пользовательских требований с помощью Visual Studio, Team Web Access, Microsoft Excel и Microsoft Project	4	2
3.	Планирование итераций: планирование спринта, оценка объема/сложности элементов работы, назначение приоритетов рабочим элементам с помощью Visual Studio, назначение приоритетов рабочим элементам с помощью веб-доступа, планирование задач спринта, оценка трудозатрат	4	3
4.	Работа с базой данных в автономном режиме: создание проекта базы данных, модификация базы данных, развертывание тестовой базы данных	4	4
5.	Разработка и анализ приложения: состав работ текущей итерации, формирование среды разработки, реализация задач, анализ кода приложения	4	5
6.	Автоматическое тестирование: общие сведения об автоматическом тестировании, создание автоматического теста, привязка автоматического теста к тестовому случаю	4	6
7.	Сценарное тестирование: разработка основных и альтернативных сценариев тестирования с помощью Visual Studio Microsoft Test Manager, документирование результатов сценарного тестирования в виде баг-репортов и руководства пользователя	4	7
	Всего часов за семестр:	28	

Также курсом предусмотрено выполнение трех практических работ. Эти работы отражают организационные процессы жизненного цикла приложения и носят вспомогательный характер. Они направлены на обеспечение рисковой устойчивости программного процесса и минимизацию издержек от его реализации. Практические работы выполняются параллельно с лабораторными работами и дополняют их. Рабочая схема выполнения практических работ представлена в табл. 1.

Таблица 2 – Рабочая схема выполнения практических работ

№ п. р.	Тема лабораторной работы: материал для изучения и характер выполняемых работ	Неделя выполнения
1.	Формирование команды и бэклога продукта: формализация требований к приложению, приоритезация требований, формирование репозитория требований	1–2
2.	Планирование релиза продукта: планирование работ по релизу продукта, покер планирования, формирование бэклога спринта и бэклога релиза, формирование репозитория задач	3–5
3.	Управление проектными рисками: идентификация, анализ и картографирование рисковых факторов на стадиях жизненного цикла приложения, выработка необходимых превентивных мероприятий для нормализации рискового фона приложения	5–7

По каждой лабораторной и практической работе оформляется отчет. Общие требования к содержанию отчетов: бумажный носитель, книжная ориентация, типовой титульный лист, тема, цель, задание (в т.ч. индивидуальное задание на разработку ПО), лаконично изложенный ход работы, минимум основных иллюстраций (копий экрана) полученных результатов, выводы по работе. **Рекомендуемый объем отчета, включая титульный лист и задание: 3 – 5 стр. для практической, 5 – 7 стр. для лабораторной работы.**

Для успешного прохождения промежуточного контроля (зачёт или экзамен) необходимо сдать и защитить все отчёты о выполнении лабораторных и практических работ, придерживаясь при этом графика их выполнения (см. табл. 1, 2).

Для выполнения лабораторных работ рекомендуется установить следующее ПО:

- Microsoft Visual Studio 2019;
- Microsoft Office 2010/2016 с расширением Microsoft Project.

Указанные программные средства можно бесплатно скачать из сети Интернет с официального сайта Microsoft. Например: http://www.visualstudio.com/ru-ru/downloads/download-visual-studio-vs.aspx.

Для полноценного использования предъявляются следующие требования к системе:

Требования к оборудованию:

- Процессор с тактовой частотой 2,2 ГГц или большей,
- ОЗУ 2 Гб или больше,
- 15 Гб доступного пространства на жестком диске.

Практическая работа №1

Тема: Формирование команды и бэклога продукта

Цель: получить практические навыки анализа потребностей заказчика программного продукта и формирования бэклога продукта для использования в скрам-процессе.

Задание

1. С использованием материалов лекций, настоящих методических указаний и материалов из сети Интернет изучить технологию Scrum для коллективной разработки приложений.
2. Провести системный анализ бизнес-процесса(ов), для которого(ых) требуется разработать программный продукт и представить описания бизнес-требований в виде User Stories.
3. Определить число и сроки спринтов, необходимых для реализации продукта, а также численность и состав скрам-команды проекта.
4. Сформулировать требования к приложению и сформировать бэклог продукта, систематизировав его во времени и пространстве важности. Построить диаграмму Use Case, отражающую структуру функциональных требований.

Практические рекомендации

Технология Scrum

В 1986 японские специалисты Hirotaka Takeuchi и Ikujiro Nonaka опубликовали сообщение о новом подходе к разработке новых сервисов и продуктов (не обязательно программных). Основу подхода составляла сплоченная работа небольшой универсальной команды, которая разрабатывает проект на всех фазах. Приводилась аналогия из регби, где вся команда двигается к воротам противника как единое целое, передавая (пасуя) мяч своим игрокам как вперед, так и назад. В начале 90х годов данный подход стал применяться в программной индустрии и обрел название Scrum (термин из регби, означающий–схватка), в 1995 году Jeff Sutherland и Ken Schwaber представили описание этого подхода на OOPSLA '95 (Object-oriented Programming, Systems, Languages, and Applications) – одной из самых авторитетных конференций в области программирования. В частности, Джефф Сазерленд предложил Scrum как преодоление классических недостатков управления проектами:

- отсутствие слаженной работы внутри команды;
- невыполнение намеченных планов
- дублирование задач внутри подразделений.

Как уже было отмечено выше, простые проекты можно реализовать любым способом, даже негибким. Scrum изначально подразумевал разработку малыми командами до 10 человек. В основе любой гибкой методики лежит принцип самоорганизации, а команда больше 10 человек самоорганизоваться уже не может. Когда больше 10 человек тогда уже начинает работать модель иерархической структуры. Сазерленд же решил что Scrum со на основе модели малой группы можно применить к сложному проекту, разбив его на подпроекты, и реализовал скрам над скрамом. Чтобы не дублировались задачи, надо корректно организовать проект.

Общее описание. Метод Scrum позволяет гибко разрабатывать проекты небольшими командами (7 человек плюс/минус 2) в ситуации изменяющихся требований. При этом процесс разработки итеративен и предоставляет большую свободу команде. Кроме того, метод очень прост – легко изучается и применяется на практике. Его схема изображена на рис. 1.1. Здесь показан цикл итераций. Итерация также называется Scrum-процесс, который также за его краткость и насыщенность называется Спринт.

Рис. 1.1. Схема Scrum-процесса

Точкой отсчета является бэклог продукта. Его формирует владелец продукта, наполняет бэклог и делает вырезку из него для предстоящего спринта. Он же решает что будет реализовано в первую очередь. Дальше команда начинает размышлять как этот спринт сформировать нормально. он совместно с командой и Scrum-мастером решает что войдет в бэклог спринта. Итерация может быть распланирована более подробно, на каждый этап ЖЦ, сумма времени этих этапов составляет время спринта.

В течение итерации планы не меняются, и этим достигается относительная стабильность разработки. Сама итерация длится от 1 до 4 недель. Она заканчивается созданием работоспособной версии продукта (рабочий продукт), которую можно предъявить заказчику, запустить и продемонстрировать, пусть и с минимальными функциональными возможностями. После этого результаты обсуждаются и требования к продукту корректируются. Это удобно делать, имея после каждой итерации продукт, который уже можно как-то использовать, показывать и обсуждать. Далее происходит планирование новой итерации и все повторяется. Внутри итерации проектом полностью занимается команда. Она является плоской, никаких ролей Scrum не определяет. Синхронизация с менеджментом и заказчиком происходит после окончания итерации. Итерация может быть прервана лишь в особых случаях.

Ролевая модель – внешнее окружение

Широко в Scrum есть всего три вида ролей: владелец продукта, Scrum-мастер и Scrum-команда (рис. 1.2). Над Scrum-процессом находится Business Owner, владелец бизнеса, а также Stake Holders – дополнительные заинтересованные лица, например, инвесторы.

Владелец бизнеса (Business Owner) – главный в предметной области, например, CEO, генеральный директор. Он взаимодействует непосредственно владельцем продукта и Scrum-мастером, например, на предмет обеспечения ресурсами либо обеспечения тестирования в реальных условиях очередного релиза. С этим предложением владелец продукта идёт к владельцу бизнеса. А Scrum-мастер занимается непосредственно обеспечением ресурсами, в том числе информационными и финансовыми, за которыми он идет прямо к руководству бизнеса. Это прямые внешние профессиональные отношения.

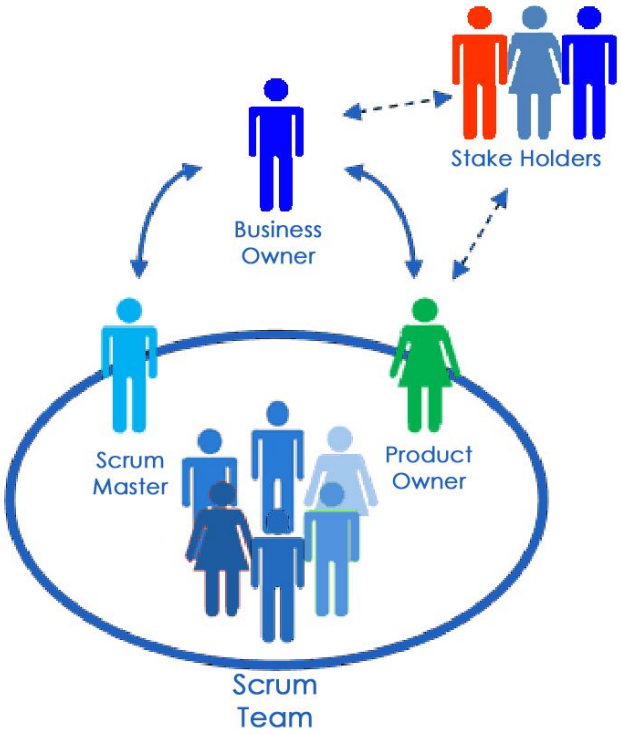


Рис. 1.2. Структура Scrum-команды и её внешнего окружения

Владелец продукта (Product Owner) – это менеджер продукта, который представляет в проекте интересы заказчика и понимает, как это продукт должен выглядеть/работать. Он обеспечивает четкое видение продукта, отвечает за общий успех разрабатываемого или поддерживаемого решения. В его обязанности входит разработка начальных требований к продукту – **Product Backlog**. Это его основной объект воздействия, который он развивает во времени с точки зрения важности (приоритетности), расставляя задачи и подзадачи (рис. 1.3). Поскольку он является представителем заказчика, то реально понимает, как этот продукт должен выглядеть и работать. Он хоть и является членом команды, но он разработкой не занимается, то есть в команде не работает над продуктом. С другой стороны он не совсем сторонний наблюдатель. Потому что он активно вовлечён в Scrum-процесс.

Формирование беклога продукта

Беклог продукта состоит из бизнес-требований, которые оформляются в виде пользовательских историй (User Stories), сценариев использования системы (Use Cases), либо в свободной форме, каждое из которых имеет свой уникальный числовой идентификатор, название, важность (ценность для бизнеса), приоритет и оценку трудозатрат. На рис. 1.1 беклог продукта представлен в левой части схемы. Развернутая структура бэклога продукта с расчетом на 3 спринта показана на рис. 1.4.

Важными обязанностями владельца продукта является планирования программного процесса – упорядочение его во времени и пространстве важности (рис. 1.3): назначение приоритетов, длительности работ, дат поставки, и пр., а также своевременное изменение требований и адаптация Product Backlog к текущей ситуации на стороне заказчика. Важно понимать, что заказчик не работает в команде, он выступает на стороне и совершенно не участвует в выполнении самой итерации. Подробно планирование спринта посредством Planning Poker рассматривается в рекомендациях к практической работе №2.

Структурирование бэклога продукта на релизы, показанное на рис. 1.4, позволяет выполнить его детализацию в формате бэклога спринта, содержащем задачи проекта, которые реализуются закрепленными за ними членами скрам-команды. Основные отличия бэклога продукта и бэклога спринта показаны в табл. 1.1.

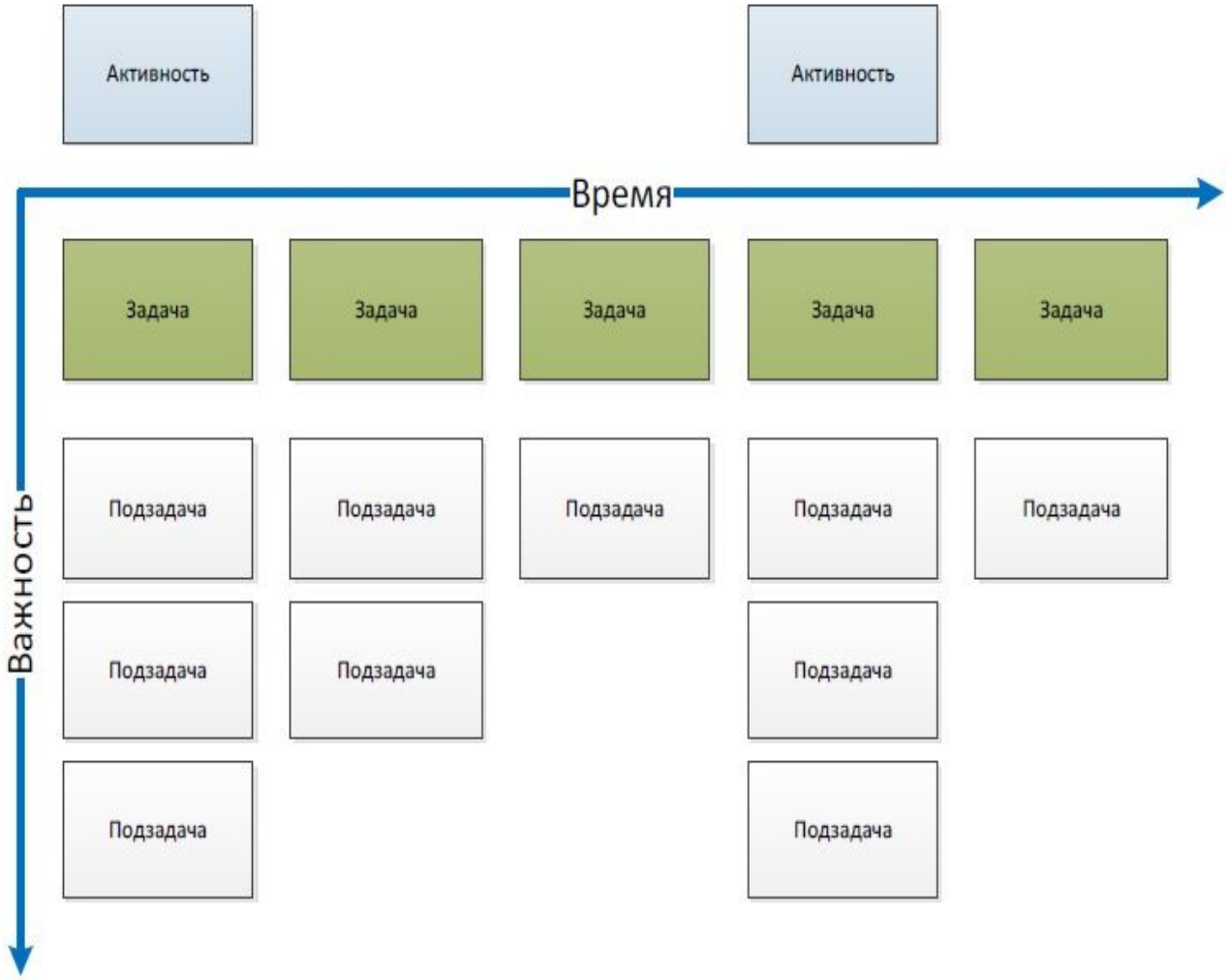


Рис. 1.3. Упорядочивание бэклога в Scrum владельцем продукта

Рис. 1.4. Приоритизация требований к продукту в Product Backlog

Таблица 1.1 – Основные отличия бэклога продукта и бэклога спринта

Показатель сравнения	Бэклог продукта	Бэклог спринта
Что входит в бэклог	Перечень всех требований, которые нужно выполнить, чтобы выпустить продукт	Список задач на спринт, структурированный по типам: подзадача, задача, фича[1], эпик[2]
Как создают элементы бэклога	Описывают бизнес-требования в общем виде (User Stories[3], Use Cases[4])	Описывают детально. Количество зависит от сложности и опыта команды. Важно спланировать бэклог так, чтобы уложиться в спринт
Кто создает бэклог	Обычно составляет владелец продукта – Product Owner	Задачи для спринта выбирает менеджер проекта или тимлид вместе с командой
Кто согласовывает и вносит изменения в бэклог	Готовый бэклог продукта согласовывают с командами разработчиков, дизайнеров, маркетологов	Вносить изменения в бэклог спринта может только владелец продукта и команда разработчиков

[1] Фича (от англ. feature – особенность, необычное свойство) – это специфическая деталь или часть продукта, которая обычно соотносится с несколькими задачами.
[2] Эпик – это большой объем работы, который декомпозируется на несколько задач и почти всегда выполняется за несколько спринтов.
[3] User Story – простое определение функциональности, которую получит пользователь как итоговую ценность.
[4] Use Case (вариант/сценарий/прецедент использования) – описание поведения продукта во взаимодействии с кем-то или чем-то из внешней среды.

У разработчиков еще есть бэклог релиза. В него включают задачи из нескольких спринтов, которые нужно решить к очередному релизу продукта. Обычно в бэклог релиза добавляют задания, которые появились после обратной связи от пользователей. Бэклог релиза составляет владелец продукта или продакт-менеджер.

Ролевая модель – внутренняя структура

Scrum-мастер (Scrum Master) обеспечивает максимальную работоспособность и продуктивную работу команды, не только в плане выполнения Scrum-процесса, но и решения хозяйственных и административных задач.

Скрам-мастер – это фронтальный руководитель. Он проводит все совещания (митинги). Он следит за соблюдением всех принципов, правил, стандартов гибкой разработки вообще и Scrum в частности. Если возникают какие-то противоречия, недоработки, он их ликвидирует, разрешает. Он защищает команду от отвлекающих факторов. Фактически это новый тип управления в самоорганизующейся команде. Команда самоорганизующаяся, по идее, должна сама организовываться, никто её не должен организовывать. Scrum-мастер организует только неосновные процессы ЖЦ, например, доставку ресурсов. Но сам скрам-процесс организуют члены команды. Никаких директив не применяется, это полностью демократическая система. Вместе с тем он является коучем (тренер) команды. Очевидно, что он является квалифицированным работником, и к нему прислушивается. Он авторитетный гибкий разработчик, грамотный специалист и профессионал, высококомпетентный работник. Он прекрасно владеет технологией Scrum и помогает членам команды понять и использовать по максимуму ценности, принципы и практики Scrum. Он помогает организовать процесс управления изменениями, решать возникающие проблемы и защищает команду от внешнего вмешательства во время итерации. Он фасилитатор, обеспечивающий успешную групповую коммуникацию. Он

способствует коммуникации, он её инициирует, проводит совещания. И следит за корректностью всех Scrum-процессов.

Scrum-команда (Scrum Team) – группа, состоящая из 5–9 самостоятельных, инициативных программистов.

Это они самоорганизованные специалисты. Им никто им не может директивно определять занятость указ, даже Scrum-мастер. Всё, что он может сделать во время спринта – это дать какие-то рекомендации, но решение принимает только команда во время спринта. Однако задаёт темп работы и мониторит следование плану спринта. Собственно команда работает самостоятельно, в процессе спринта команда самоорганизуется. Команда состоит из универсальных разработчиков, они кросс-функциональны, то есть обладают всеми необходимыми навыками для разработки продуктов в любой из ролей. Распределение по ролям происходит по умолчанию, команда самоорганизуется. Если есть объективные причины для перехода на другую роль, то на ежедневных собраниях это обсуждается.

У команды разработки нет подкоманд, которые выполняли бы отдельные функции тестирования или бизнес-анализа. Отдельные члены команды могут владеть специализированными знаниями в различных областях, Junior-специалисты могут быть вообще не компетентными. Но ответственность лежит на всей команде в целом, это называется коллективная ответственность. Это один из базовых принципов групп равных, к которым, несомненно, относятся Scrum-команды – все несут в равной мере ответственность за всё, что они делают. Это справедливо для группы равных, потому что они равны в своих правах и своих обязанностях. Даже если стажер приходит на работу, он должен понимать, что хоть он и стажёр и почти ничего не умеет и пока мало что знает, но он уже вошёл в эту команду и значит в равной степени отвечает за продукт. Это сразу возлагает на него серьезный груз ответственности и заставляет его в некоторой степени серьезно воспринимать эту объективную действительность: «Все работают, и я тоже должен работать, я работаю как могу, не так эффективно, как все, может даже в разы менее эффективно, но тем не менее, я отвечаю за продукт, как и все». Такая ответственная позиция его очень быстро подтянет на общий уровень, менее чем за год. В таких обстоятельствах профессиональное созревание резкого ускоряется, это такой бизнес-инкубатор. Он должен с энтузиазмом выполнять свои обязанности, тогда созревание в этом инкубаторе может быть довольно-таки быстрым и ускоренным без ущерба для качества. Быстрее, чем это возможно, нейронные связи в коре головного мозга не образуются. Если человек гений, у него может быть всё гораздо быстрее, но гении не так часто встречаются. На уровне нормальной физиологии профессиональная адаптация составляет год-полтора-два, но бывает и быстрее.

Спринт – это своего рода «скороварка», где команда запирается на несколько недель. И там возникает высокое давление, повышается температура, в данном случае это социальное давление и социальная температура, рост которых способствует повышению эффективности труда на 10-15%, что достаточно много. В такой ситуации всё происходит эффективнее, и созревание, как продукта, так и членов команды, происходит быстрее.

Размер команды разработчиков оптимально 7±2 человек. При численности менее трех человек снижается производительность. К тому же наиболее глубокие противоречия возникают именно в команде из двух человек. Более 10 человек – это уже трудности в управлении, способность к самоорганизации начинает резко снижаться.

Состав команды в Scrum показан в нижней части рис. 1.1. Помимо нескольких программистов и тестировщиков в команде может выделяться системный аналитик. Он подключается на начальных этапах разработки, он анализирует backlog и бизнес-требования, преобразует в требования функциональные, согласовывает их с владельцем продукта и формирует backlog спринта. Иногда требуется привлечение таких ролей, как системный администратор, архитектор, верстальщик, дизайнер и прочие специалисты, что определяется спецификой проекта. Однако следует понимать, что все эти роли объединяют в себе разработчики Scrum-команды.

Недостатки технологии Scrum

Прежде всего, для реализации Scrum требуются навыки самоорганизации на очень высоком уровне. Это гибкая методика, группа равных, а значит, равная коллективная ответственность и необходимость эффективной самоорганизации. Способность работать без специальных методов мотивации вроде пресловутых кнута и пряника. Не ради денег, не для получения результата, а просто работать для дела. Не то чтобы совсем не надо ориентироваться на результат, есть цель, и она должна быть оформлена как метафора (одна из практик XP, фиксирующая, что суть проекта должна умещаться в 12 емких фразах или в некотором образе). Необходимо помнить, что *основной целью использования любых гибких методологий является создание предсказуемого процесса разработки ПО заданного качества*. Поэтому работать надо не для достижения результата любой ценой, а для реализации качественного процесса. Поэтому важен очень высокий уровень самоорганизации.

В Scrum-процессе оптимистичные оценки могут превалировать над реальными – это тоже недостаток Scrum, он вообще в целом очень оптимистичен с точки зрения организации процесса. Это может несколько отодвигать команду от реальности, тем более, что она в этом котле с высоким давлением действительно может в отрыве от реальности работать. Поэтому Scrum-мастер должен следить за тем, чтобы команда не выходила за рамки реальности и не погружалась чрезмерно глубоко в свой оптимистичный сказочный мир.

Отсутствие планирования может приводить к рискам в узких местах. Например, это может выразиться в нехватке каких-либо ресурсов. За этим тоже должен следить Scrum-мастер, он должен стремиться минимизировать все недостатки, равно как и все риски, связанные с этими недостатками. Здесь имеется в виду отсутствие планирования в течение спринта, потому что перед спринтом в рамках практики Sprint Planning Meeting планирование всё-таки проводится, причем двухступенчатое. Внутри спринта всё идет по изначальному плану текущего спринта. От этого плана могут отступить, так как методика гибкая и поэтому согласно четвертому постулату гибкой разработки никто не следует жёсткому плану. Подробно двухступенчатое планирование спринта посредством Planning Poker рассматривается в рекомендациях к практической работе №2.

Внешнее окружение может отрывать команду по предыдущим решениям, по срочным доработкам, по параллельным проектам. Такое часто случается и это вредит проекту. Поэтому, задача Scrum-мастера – заблокировать все эти транзакции и выдергивать оттуда специалистов только по очень большой нужде, когда никак нельзя отказать, когда авария и необходимо срочно вмешаться, срочно доработать продукт. В этом экстраординарном случае он может кого-то извлечь из спринта и на Daily Scrum Meeting об этом заявить, при необходимости инициировав перестройку ролевой модели. Как правило, Scrum-команда тогда немного перегружается, но от плана не отступает. При длительном отрыве работника спринт может оказаться под угрозой проблем, так как члены Scrum-команды работают с перегрузкой. Они могут получить сверхурочные деньги за срочную работу, главное, чтобы работа шла по плану.

Не понятна критичность нарушения спринта или отдельных частей его доработок, так как нет критического пути для решения задач. Не обязательно, чтобы критического пути не было совсем. В принципе, он может быть прочитан Scrum-мастером, либо при планировании спринта может как-то неявно обозначаться. Всё-таки здесь работают профессионалы и у них в головах у всех есть своя версия критического пути, которые согласуются на Daily Scrum Meeting, а значит, что у Scrum-команды есть понимание того, что и когда надо делать. Это может подвести, эти оптимизационные радужные вещи, вот эти оптимистические оценки могут немножко нарушить процедуру адекватного восприятия действительности. Тем не менее, люди здесь качественно работают обычно, поэтому проблем с этим обычно не бывает.

Так или иначе, все недостатки Scrum-процесса – это зона ответственности Scrum-мастера, который должен это всё отслеживать и нивелировать просвечивающиеся недостатки, ликвидируя связанные с ними проблемы преимущественно на корню.

Практическая работа №2

Тема: Планирование релиза продукта

Цель лабораторной работы: получить практические навыки краткосрочного и среднесрочного планирования работ по программному проекту для разработки релиза продукта и выполнения мелких доработок.

Задание

1. Провести Sprint Planning Meeting. Проанализировать бэклог продукта, сформированный при выполнении практической работы №1. Для каждого функционального требования определить одну или несколько проектных задач и соответствующих им подзадач. Результаты представить в формате таблицы 2.1

Таблица 2.1 – Структура беклога продукта в контексте проектных задач

Номер и наименование задачи	Приоритет задачи	Номер и наименование подзадачи	Приоритет подзадачи

2. Сформировать беклог спринта. Разыграть покер планирования и реализовать двухступенчатую процедуру планирования спринта:

2.1. Для каждого элемента беклога спринта определить тип (подзадача, задача, фича, эпик), зафиксировать покерный расклад на каждой итерации оценивания. Для итераций, в результате которых консенсус не достигнут, привести описание причин перехода к следующей итерации оценивания. Результаты представить в виде таблицы 2.2.

Таблица 2.2 – Структура оценивания беклога спринта

Номер и наименование элемента	Тип элемента	Номер итерации	Покерный расклад	Описание причин перехода к следующей итерации оценивания

2.2. На основе данных табл. 2.2 для каждого элемента беклога спринта определить трудоемкость (в стори поинтах или идеальных человеко-часах), время, затраченное на оценивание (в минутах), исполнителей из числа членов скрам-команды и срок реализации (период или дедлайн). Формат беклога спринта показан в табл. 2.3.

Таблица 2.3 – Структура беклога спринта

Номер и наименование элемента	Приоритет элемента	Трудоемкость элемента	Ответственные работники	Время оценивания	Сроки реализации

3. По ходу работы над релизом продукта и в процессе обсуждений на Daily Scrum Meeting и Sprint Review Meeting сформировать бэклог релиза. Особое внимание уделить сложным элементам типа «фича» и «эпик».

Практические рекомендации

Scrum-практики

Первой задачей команды является постановка для итерации реально достижимых и приоритетных для проекта в целом задач (на основе Project Backlog и при активном участии

владельца продукта и Scrum-мастера). Второй задачей является реализация поставленных задач во что бы то ни стало, в отведенные сроки и с заявленным качеством. Важно, что команда сама участвует в постановке задачи и сама же ее выполняет. Здесь сочетается свобода и ответственность, подобно тому, как это организовано в MSF. Здесь же "просвечивает" дисциплина обязательств.

Рассмотрим структуру одной Scrum-итерации, показанной на рис. 1.1. Заказчик формулирует бизнес-требования и вносит их в бэклог проекта, откуда делается вырезка в рамках планирования спринта. Sprint Planning Meeting проводится в начале каждого Sprint. Сначала Product Owner, Scrum-мастер, команда, а также представители заказчика и пр. заинтересованные лица (фактически, все субъекты, отраженные на рис. 1.2) определяют, какие требования из Project Backlog наиболее приоритетные и их следует реализовывать в рамках данного Sprint. Формируется Sprint Backlog, включающий задачи, обязательства по выполнению которых за спринт принимает на себя команда. Каждая задача оценивается в идеальных человеко-часах. Решение задачи не должно занимать более 12 часов или одного дня. При необходимости задача разбивается на подзадачи

Далее Scrum-мастер и Scrum-команда определяют то, как именно будет реализован этот объём работs из Sprint Backlog. Для каждого элемента Sprint Backlog определяется список задач и оценивается их трудоемкость, выполняется разложение на задачи в соответствии с логикой, показанной на рис. 1.3. Здесь задействованы аналитик и архитектор. Между спринтами может потребоваться перепланирование, так как может измениться архитектура разрабатываемого продукта.

Продолжительность Sprint Planning Meeting ограничена сверху 4-8 часами в зависимости от продолжительности итерации, опыта команды и т. п. Если команда опытная – меньше разговоров, больше дела. Неопытная команда может совещаться дольше. В пределе – это весь рабочий день – 8-ми часовое совещание – это, конечно, очень серьёзное мероприятие, надо делать перерывы. Поэтому лучше меньше, да лучше – можно и за пару часов договориться. На практике обычно Sprint Planning Meeting продолжается около двух часов. Даже если проект простой, то нужно минимум 40 минут на решение всех вопросов.

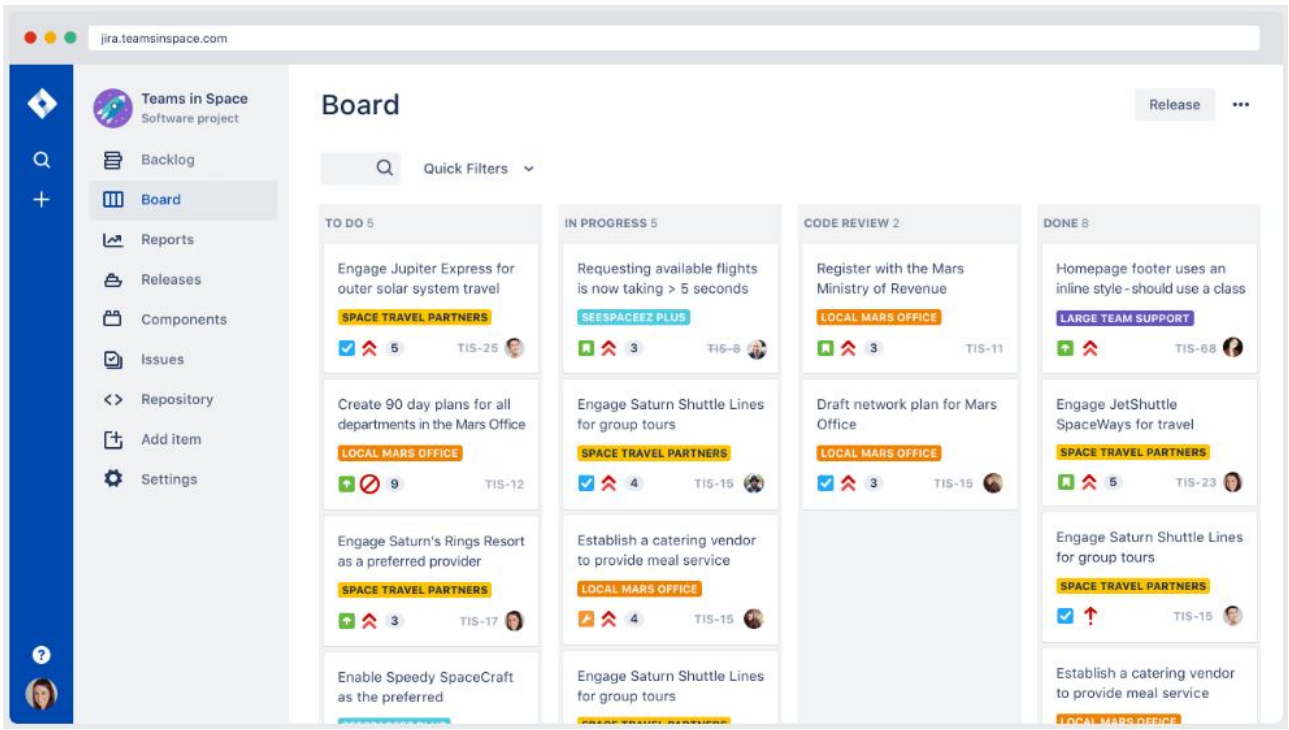
Далее начинается собственно спринт продолжительностью 1–4 недели, где каждый день происходит Daily Scrum Meeting – каждодневное пятнадцатиминутное совещание, целью которого является:

- достичь понимания того, что произошло со времени предыдущего совещания;
- скорректировать рабочий план сообразно реалиям сегодняшнего дня;
- обозначить пути решения существующих проблем.

Daily Scrum Meeting должен начинаться точно вовремя. Это дисциплина, а там где есть дисциплина, там порядок. Где порядок, там возможны действительно стоящие и ценные результаты. Если порядка в процессах деятельности нет, то результат будет ущербным. Порядок определяет качество процесса, а качественный процесс определяет качество результата. Поэтому дисциплина – это залог качества. Поэтому опознаний быть не должно, всё нужно начинать точно вовремя, это уважение ко времени, это уважения к другим членам команды, это уважение к себе, в конце концов. Это то, чем должен руководство каждый работник – он должен уважать себя, он должен уважать других своих коллег. *Уважение* – это одно из базовых правил взаимодействия в команде, правило, определяющее первый постулат гибкой разработки (Люди и их взаимодействие важнее процессов и инструментов) и один из основных принципов организации процесса по XP. Кратковременность совещания не позволяет тратить много времени на подготовительные работы к его проведению. Обычно оно проводится стоя, в течение 15–20 минут.

Daily Scrum Meeting проводится в одном и том же месте в течение спринта. Это тоже важно, потому что в этом месте организована знакомая обстановка, там стоит Scrum-доска (рис. 2.1, а), на которую наносится схема бэклога спринта.

а)



б)

Рис. 2.1. Scrum-доска: а) физическая, б) электронная в Jira

Scrum-доска – одна из самых гибких технологий, применяемых для визуализации групповой работы над проектами. Данная технология активно применяется в различных сферах современной жизни для четкой систематизации работы.

Scrum-доска похожа на многоколоночный список элементов, позволяющий команде разработчиков:

- отслеживать все задачи, которые необходимо решить (To Do), работая над текущим проектом;
- контролировать процесс работы по текущим задачам (In Progress), наблюдать за сотрудниками и грамотно распределять между ними задачи/функции;
- следить за прогрессом текущего спринта, чтобы задачи вовремя попадали в список выполненных (Done).
- проводить аналитические совещания с целью обсудить успехи компании.

Основное преимущество Scrum-досок – выявление проблем. Не получится забыть про задачу, потерять ее или запутаться в обилии дел, не понимая, чем конкретно сейчас нужно заниматься. Четкая структура Scrum-доски «очищает» голову разработчиков и позволяет им сконцентрироваться на написании кода, а не на менеджменте.

Очень хорошо, если есть система управления Scrum-процессом, такая как, например, Azure Devops Server или Jira, где Scrum-доска реализована в электронном виде (рис. 2.1, б). Такая доска синхронизируется по сети. Данные доступны всем работникам независимо от их месторасположения, используемого устройства или используемой сети.

Лучшие программы для создания цифровых Scrum-досок:

Jira – безусловный лидер, уже ассоциирующийся с Agile-разработкой. Поддерживает все необходимые для разработчиков функции, встроенную систему аналитики, множество механизмов сортировки данных и коммуникацию между сотрудниками.

ClickUp – продвинутый цифровой scrum-board с возможностью наглядно устанавливать цели, а также находится в постоянном контакте со всеми программистами компании. Комментировать каждую карточку, например.

monday.com – удобная система взаимодействия в Agile-команде с функцией интеграции других популярных сервисов. Также monday.com любят за обилие готовых шаблонов со всеми необходимыми колонками.

Wrike – популярный инструмент для менеджмента больших команд в ходе спринтов. Им пользуются крупные корпорации в духе Google и Airbnb. Wrike более универсален и адаптирован не только под решение задач разработчиков.

В процессе совещания каждый участник Scrum-команды отвечает на три вопроса:

1. Что я сделал со времени предыдущей встречи для того, чтобы помочь команде достигнуть цели Спринта?
2. Что я буду делать до следующей встречи для того, чтобы помочь команде достичь цели Спринта?
3. Вижу ли я проблемы и препятствия для себя или команды, которые могли бы затруднить достижение цели Спринта?

Если эти проблемы не удаётся решить за 15 минут, то далее над их решением работает Scrum-мастер. Он присутствует на совещании, но решение принимает команда. Скрам мастер решает проблемы и в следующий митинг он может прийти и объявить о результатах своих решений. При этом участники совещания не тратят время на любезности, так как в данном формате взаимодействия это ни к чему. В профессиональной среде обычно стремятся обойтись без любезностей, потому что на это уходит очень много времени и психической энергии. Это не значит, что не надо быть вежливым, никто не отменял воспитанности, но лишние любезности обычно оставляют за скобками и считают их подразумевающимися. Совещание в положении стоя также не располагают к длительным обсуждениям.

В Daily Scrum Meeting может принимать участие любое заинтересованное лицо, но только участники Scrum-команды (фигуры внутри эллипса на рис. 1.2) имеют право принимать решения. Правило обосновано тем, что они давали обязательство реализовать цель итерации, и только это дает уверенность в том, что она будет достигнута. На них лежит ответственность за их собственные слова, и, если кто-то со стороны вмешивается и принимает решения за них, тем самым он снимает ответственность за результат с участников команды. Такие встречи поддерживают дисциплину обязательств в Scrum-команде, способствуют удержанию фокуса на целях итерации, помогают решать проблемы "в

зародыше". Ближе к концу спринта реализуется тестирование. Здесь также нужны аналитик и архитектор для поддержки написания сценариев тестирования.

В случае, если по проекту работает несколько команд, то реализуется практика Scrum of Scrums (скрам над скрамом). Проводится после Daily Scrum Meeting и позволяет командам обсуждать работу, фокусируясь на общих областях и взаимной интеграции. Повестка та же, что и на Daily Scrum Meeting плюс следующие вопросы:

4. Что каждая команда сделала с момента предыдущего ежедневного совещания?
5. Что каждая команда сделает к следующему ежедневному совещанию
6. Есть ли проблемы, мешающие или замедляющие работу каждой команды?
7. Нужно ли другой команде сделать что-то из задач вашей команды?

То есть после Daily Scrum Meeting собираются Scrum-мастера соответствующих команд и обсуждают проект в целом. Scrum-мастер владеет полной информацией о своем подпроекте и может её транслировать от лица команды. Он является выразителем мнением команды, поэтому его одного достаточно на таком совещании. И если грамотно построить эту работу и обеспечить корректную интеграцию подпроектов в единый проект и системное тестирование, то можно рассчитывать на качественный результат.

По завершении спринта выпускается очередная версия, которая презентуется заказчику, а также выполняется ретроспективный анализ спринта в рамках Sprint Review Meeting, который проводится в конце каждого спринта. Встреча ограничена четырьмя часами в зависимости от продолжительности итерации и прироста функциональности продукта.

Сначала Scrum-команда демонстрирует Владельцу продукта (Product Owner) сделанную в течение Sprint работу, а тот в свою очередь ведет эту часть митинга и может пригласить к участию всех заинтересованных представителей заказчика. Привлекается максимальное количество зрителей из числа заинтересованных лиц. Команда демонстрирует прирост функциональности продукта. Все члены команды участвуют в демонстрации (один человек на демонстрацию или каждый показывает, что сделал за спринт). На практике кто-то один или 2 человека из Scrum-команды делегируется как спикер и всё докладывает. Делается доклад с презентацией, при этом кто-то может что-то дополнить. Поскольку это гибкая методика, то могут быть разные варианты и схемы. При этом нельзя демонстрировать незавершённую функциональность – это важное правило. Это не только правило Scrum, это вообще правило психологии. Если хотите, чтобы вас воспринимали всерьёз, не показывайте никому не доделанную работу. Пускай довольствуются завершенным вариантом. Недоделанная задача акцентирует внимание на том, что недоделано. Значит что-то здесь не так, раз она не доделана и презентуется. Когда демонстрируется готовая к использованию вещь, это оказывает хороший психологический эффект. Недоделанная работа может произвести благоприятное впечатление только на специалистов, когда они могут оценить объем проделанной работы. Заказчика же интересует результат, поэтому на Sprint Review Meeting следует демонстрировать только завершенные варианты текущей версии продукта. Product Owner при этом определяет, какие требования из Sprint Backlog были выполнены, и обсуждает с командой и заказчиками, как лучше расставить приоритеты в Sprint Backlog для следующей итерации.

Во второй части митинга производится анализ прошедшего спринта, который ведет Scrum-мастер. Scrum-команда анализирует в последнем Sprint положительные и отрицательные моменты совместной работы, делает выводы и принимает важные для дальнейшей работы решения. Scrum-команда также ищет пути для увеличения эффективности дальнейшей работы. Затем происходит переход в начало на планирование и цикл повторяется.

Завершается разработка, когда backlog опустел. В этом случае перед заключительным Sprint Review Meeting в течение 1-2 дня Scrum-команда выполняет регрессивное, нагрузочное и, возможно, системное тестирование, после чего осуществляется демонстрация завершенного продукта.

Планирование спринта посредством Planning Poker

Планирование спринта – это регулярный процесс командного обсуждения каждой задачи спринта. Один из распространенных способов коллективной оценки сложности задач спринта – Planning Poker (покер планирования). Подход позволяет обсудить оценки, данные экспертами, и прийти к компромиссу. Покер планирования основан на методе Wideband Delphi, который начали использовать в 1970-х годах. В 2002 году этот метод был адаптирован для Agile-команд Джеймсом Гренингом, одним из участников разработки Agile-манифеста. Метод применяется Agile-командами, организующими свою работу по Scrum, хотя изначально появился в подходе XP.

Независимо от числа участников встречи, формата её проведения (онлайн или офлайн) и особенностей разрабатываемого продукта совещания по покерному планированию обычно состоят из следующих шагов.

Шаг 1. Организация встречи. Собраться можно как в офисе, так и удаленно. Для второго случая существуют разнообразные онлайн-сервисы с функционалом для Planning Poker, например, Jira или онлайн-сервис pplanning.ru. Для офлайн-встречи необходимо заранее подготовить колоду карт на всех участников.

Важно ограничить встречу по времени и строго следить за таймингом. Обычно на одну сессию выделяется 2–4 часа в зависимости от объема обсуждаемых задач, количества участников и уровня их вовлеченности.

В некоторых командах в Planning Poker участвуют все сотрудники, даже если у некоторых недостаточно экспертности в оцениваемой задаче. Однако подход будет более эффективным, если в оценке каждой задачи будут принимать участие только те участники команды, кто потенциально может выполнить задачу. Именно они смогут высказать экспертное мнение, которое сделает оценку более обоснованной и точной.

Заказчик (постановщик задачи – менеджер продукта, маркетолог или даже генеральный директор) обязательно должен присутствовать на планировании, чтобы объяснить суть задачи, рассказать, как он ее понимает, донести до команды почему задача важна и мотивировать исполнителей на ее выполнение. Цель инженерной команды – за счет дополнительных вопросов выяснить максимальное количество подробностей, вытянуть из заказчика скрытые требования, предложить свои идеи по реализации и выработать наилучший способ решения. Либо объяснить, почему выполнение задачи стоит отложить на некоторое время или не брать в работу вообще.

Шаг 2. Подготовка к процедуре оценивания. Оценивание производится с помощью цифровых карт — это и есть так называемый Planning Poker: все участники команды исполнителей должны в закрытую оценить сложность задачи. Для этого участникам оценки раздается одинаковое количество карт с цифровыми значениями (рис. 2.2). Команда договаривается, в каких единицах измерения будут делать оценки задач. Оценка задач может осуществляться в Story Point’ах или в идеальных часах.

Story Point – единица измерения в подходе Scrum, в которой оцениваются усилия команды на выполнение задачи.

Идеальный час – рабочий час, во время которого сотрудник 100% занят выполнением задачи без перерывов.

Карта с цифрой 1 обозначает оценку в 1 идеальный час, если команда договорилась оценивать задачи в идеальных часах, и 1 Story Point, если оценки сложности задач производятся в Story Point’ах. Карта с цифрой 2 — это 2 идеальных часа или 2 Story Point’а и т. д.

Кроме цифр, на карточках могут быть и другие символы, например:

- «?» – затрудняюсь ответить, требуется уточнение данных;
- «∞»или «Pass» – невыполнимая задача;
- «о» или «0» – очень простая или уже готовая задача;
- «π» или чашка кофе – требуется перерыв (“I’m tired and want some pie”).

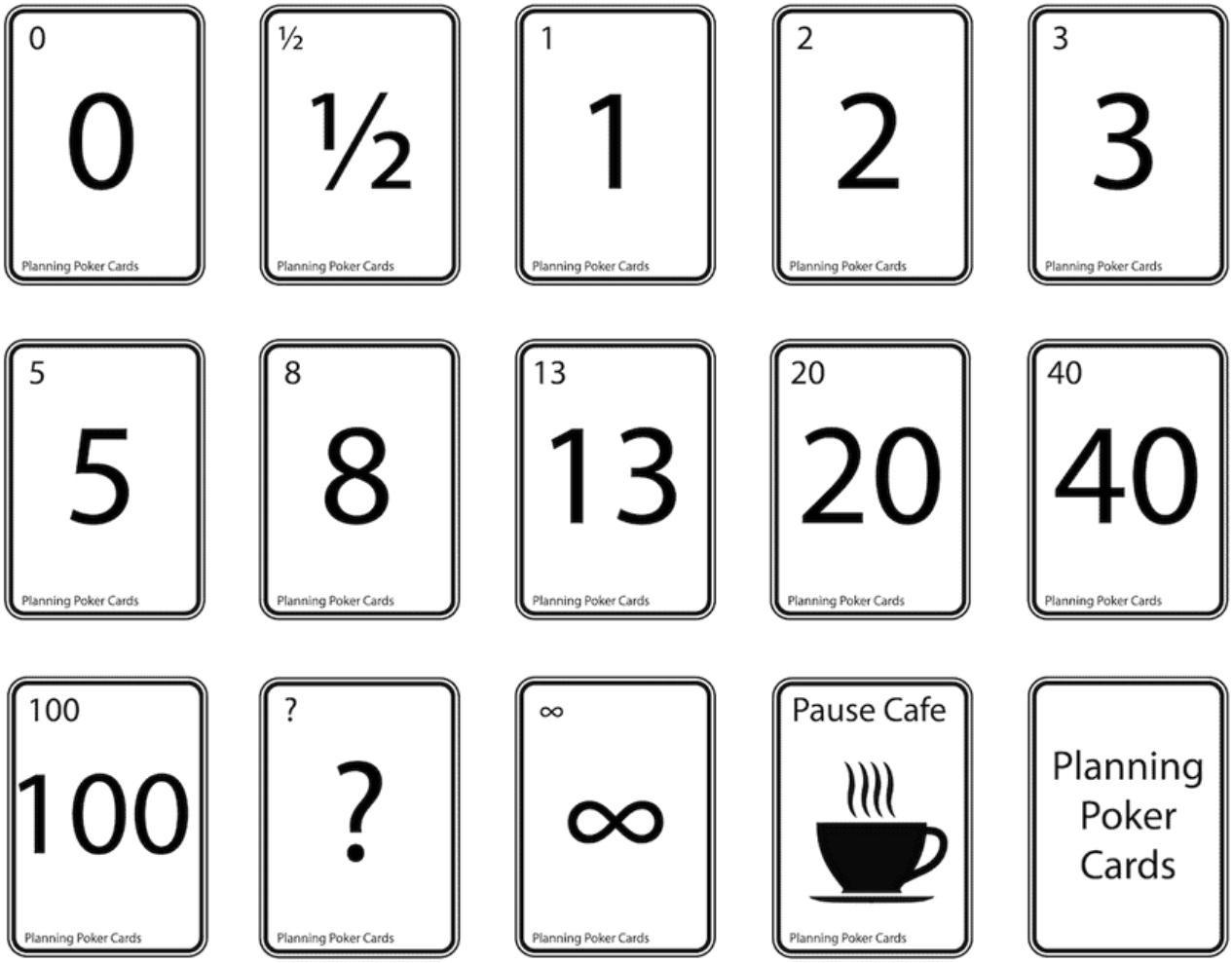


Рис. 2.2. Специальная колода карт Planning Poker

Важно, чтобы все участники договорились о значениях всех карт и правильно понимали их суть перед началом оценивания.

Шаг 3. Выбор задачи из бэклога и подготовка к оценке. Команда вытягивает из бэклога продукта элемент с самым высоким приоритетом. Заказчик зачитывает элемент (это всегда делает строго заказчик, чтобы избежать «сломанного телефона» при передаче данных и ускорения процесса выработки решения), объясняет, что именно нужно сделать и почему это важно.

Цель каждого участника команды исполнителей, задавая различные вопросы заказчику и членам команды, выяснить что именно нужно сделать и понять, как лучше всего решить задачу. Менеджер продукта как можно точнее отвечает на их вопросы. Затем исполнитель объясняет, что конкретно он планирует сделать, и уточняет, устроит ли такой вариант заказчика и команду. Все участники высказываются по очереди, и после того, как все определились с наилучшим решением, наступает этап оценки сложности задачи.

Шаг 4. Оценивание. Каждый участник выбирает карту со значением, которое, по его мнению, соответствует степени сложности задачи. Чтобы оценка была независимой, а сотрудники не привязывались к мнению друг друга, карты кладутся на стол рубашкой вверх.

После того как все участники проголосовали картами, модератор просит перевернуть их, чтобы были видны цифры с оценками. Далее он предлагает объяснить свой выбор двум участникам: тому, кто выбрал наименьшее значение, и тому, кто выбрал наибольшее значение. Например, оценивая сложность задачи, сотрудник А выбрал карточку с цифрой 3, а сотрудник Б — с числом 13. Все остальные эксперты выбрали карточки между этими значениями. В такой ситуации полезнее всего сначала услышать мнение именно тех

сотрудников, чьи мнения разошлись сильнее всего. Тот, кто поставил наименьшую оценку (сотрудник А), делится с командой, как именно он планирует выполнить задачу так быстро. Сотрудник А мог выбрать наименьшую цифру по нескольким причинам:

- он выбрал самый простой и изящный способ решения задачи;
- он неверно понял суть задачи;
- он не знает или забыл о какой-либо информации по задаче, которая усложнит ее выполнение.

Участник, который дал самую большую оценку (сотрудник Б) должен рассказать о том, какие риски и сложности он предвидит при выполнении этой задачи. Он мог выбрать наибольшую цифру также по ряду причин:

- он выбрал самый сложный и трудоемкий способ реализации задачи;
- он учел риски, которые не увидели остальные участники.

После повторного обсуждения с учетом всех подводных камней, команда принимает решение о том, какая оценка является более подходящей для задачи. Этот срок заносится в карточку с описанием задачи в таск-менеджере. Как правило, после такого обсуждения двух крайних оценок участники договариваются о лучшем способе реализации задачи.

Таким образом, **главная цель оценивания сложности** не предсказать, когда задача будет готова, а убедиться, что все участники одинаково понимают задачу. Если один участник оценивает задачу в ? стори поинта, а другой в 3 стори поинта, они явно задумали выполнять задачу по-разному, и поэтому должны согласовать свои действия и объяснить, почему надо делать именно так, как они думают. Иногда расхождение в оценке может быть обусловлено разным опытом в решении схожих задач, в этом случае берется максимальная оценка, но если срок отличается более чем на 1 день, то задача выполняется в парном программировании (одна из практики методики XP), когда тот, кто оценил в меньшую сторону, руководит тем, кто оценил в большую.

Если консенсус не достигнут, происходит второй тур голосования. Участники снова вытягивают карточки, кладут их рубашкой вверх, после чего карты вскрываются и участники видят новый диапазон из оценок. После второго тура голосования этот диапазон должен уменьшиться.

Шаг 4 может повторяться до достижения консенсуса. В результате команда приходит к общему видению способа реализации задачи и оценке ее сложности или объема работ. Либо после второго тура команда выбирает среднее значение из полученного диапазона или останавливается на варианте, близком к пессимистичной оценке. В любом случае выбранное значение трудозатрат фиксируется, а команда переходит к следующей задаче, и шаги 3 и 4, повторяются.

Шаг 5. Ретроспектива. После завершения спринта команды, работающие по Scrum, проводят ретроспективу. Один из вопросов, которые можно рассмотреть на ретроспективе – это задачи, по которым полученные при планировании оценки сильно отклонились от фактических значений. Например, если команда сошлась во мнении, что на задачу уйдет 6 идеальных часов, а исполнитель затратил на нее в 3 раза больше времени, стоит обсудить, из-за чего так случилось, какие возникли трудности в выполнении задачи и как можно учесть в будущем полученный опыт.

Способы повышения вовлеченности команды в Planning Poker

Благодаря игровой составляющей Planning Poker изначально имеет высокие шансы вовлечь всех участников в процесс. Заинтересоваться и выставлять наиболее корректные оценки команде также помогают:

- грамотные вопросы от Scrum-мастера. Важно, чтобы сотрудники не ощущали осуждения за свой выбор, например, за слишком высокие или низкие значения. Уточняющие вопросы о том, почему выбрана именно это карта, помогут участникам раскрыть свою точку зрения и настроят на позитивное восприятие процесса;

– карточки без значений. Например, символы с просьбами о перерыве или уточнениях делают процесс более гибким, а условия более мягкими, позволяют участникам в любой момент сделать паузу или получить дополнительные данные;

– калибровка оценок. Метод не всегда дает ожидаемые результаты с первых попыток: иногда требуется несколько месяцев таких регулярных встреч. С каждым разом команда будет всё более точной в оценках, адаптируясь к процессу;

– обсуждение запланированных и фактических оценок. Сотрудники больше вовлекаются в процесс, если вместе с ними разобраться в причинах отклонений от плана без каких-либо осуждений и попытаться совместно найти пути решения возникающих затруднений.

Особое внимание следует уделить первой игре в Planning Poker.

1. Эксперты. Перед планированием стоит удостовериться, что на каждую оцениваемую задачу есть как минимум два сотрудника, имеющих навыки и знания для ее выполнения.

2. Тайминг. Стоит запланировать для встречи строгие временные рамки и следить, чтобы команда укладывалась в них. Обычно сессия занимает несколько часов. Инициатор должен напоминать коллегам о сути встречи, чтобы она проходила продуктивно и не затягивалась.

3. 2 тура. Оценку одной задачи стоит проводить не больше, чем в два тура. Если после обсуждения полярных значений и повторного выбора значений числовой диапазон остался слишком широким, скорее всего, задача слишком крупная и ее следует разделить на более мелкие.

4. 1 день. На выполнение каждой задачи должно потенциально отводиться не более одного дня. Если приходит понимание, что работа займет больше времени, её следует делить на несколько мелких задач.

Преимущества и недостатки Planning Poker в Agile.

Agile-оценка – жизненно важный навык для проектных групп, стремящихся получить высококачественные результаты в ожидаемые сроки. Овладев искусством Agile-оценки и организовав игру в Planning Poker онлайн, удаленные команды могут устанавливать реалистичные ожидания, эффективно расставлять приоритеты задач и развивать сотрудничество внутри команды.

Ключевым недостатком считается необходимость присутствия как минимум двух человек, которые имеют компетенции для выполнения каждой из обсуждаемых задач. Planning Poker не даст положительного эффекта, если оценки дают сотрудники, не имеющие экспертизы и навыков для выполнения оцениваемых задач.

Преимущества подхода:

– участники получают возможность уточнить любые неясные моменты по задачам, что сократит в будущем возможность ошибок при выполнении работы;

– команда обсуждает разные алгоритмы реализации задач и выбирает оптимальный, обмениваясь опытом и знаниями;

– в процессе обсуждения сотрудники выявляют возможные риски и могут заложить дополнительное время на риски;

– участники лучше узнают о сути и нюансах работы коллег, проникаясь общей идеей проекта.

Практическая работа №3

Тема: Управление проектными рисками

Цель: освоение методики идентификации, анализа и картографированя рисковых факторов на стадиях жизненного цикла ИС, выработка необходимых превентивных мероприятий для нормализации рискового фона процесса информатизации.

Задание

1. Идентифицировать риски проекта в категориях разработки, внедрения и применения продукта. Риски разработки также следует искать среди задач со значительными временны́ми характеристиками (трудоемкость, время оценивания, сроки).
2. Провести оценивание рисков, используя следующие две ранговые шкалы: 1) вероятности реализации риска и 2) существенности последствий риска.
3. Заполнить карту рисков, используя полученные результаты их идентификации и оценивания. Провести ранжирование рисков.
4. На основе данных табл. 3.1–3.5 составить перечень мероприятий для нормализации рискового фона объекта информатизации.

Практические рекомендации

Методика риск-менеджмента программного процесса

Любая организация, являясь элементом нестабильной социально-экономической среды общества, испытывает необходимость перманентного управления рисками. Важным фактором эффективности здесь является рисковый фон, определяющий потенциал успешности функционирования объекта информатизации. Исследование рискового фона позволяет идентифицировать основные риски, определить их возможные последствия, предложить превентивные подстроечные мероприятия по купированию возможных негативных последствий рисков.

Понимая риск как степень неопределённости функционирования объекта информатизации в ситуации, когда имеется возможность отклонения результатов от предполагаемой цели, следует также тщательно проанализировать область применения ИС. При этом необходимо выявить риски, связанные с низкой эффективностью принятия решений и/или высокой субъективностью результатов реализации принятых управленческих решений.

Идентификация рисков

В подразделе 1 следует разделить риски на 3 категории, как показано в табл. 3.1. Необходимо выявить 3–5 рисков каждой из трех категорий, актуальных для проектируемой ИС. Это должны быть риски, природа которых наиболее очевидна. Примеры некоторых рисков показаны в табл. 3.1. Риски применения (категория «П») чаще носят сугубо частный характер сообразно области применения. Например, риск нарушения хода учебных занятий в соответствии с индивидуальными образовательными траекториями обучающихся.

Таблица 3.1 – Классификация рисков для решения поставленной задачи, категории рисков и примеры рисков по категориям

Класс риска	Подкласс рисков	Примеры рисков с указанием шифра
Риски разработки ИС для обеспечения успешности	–	Р-1 – увеличение нагрузки на работников объекта на этапе его обследования

функционирования объекта (категория «Р»)		Р-2 – некорректные требования к ПО, сформулированные заказчиком Р-3 – недостаточное финансирование проекта
Риски внедрения ИС и её применения для управления объектом (выработки и применения управленческих решений по приведению объекта в заданное состояние)	Риски нарушения функционала реализации разработанных решений – системные риски (категория «С»)	С-1 – загрузка недостоверных данных для анализа, либо их отсутствие С-2 – нестабильная работа сети Интернет С-3 – недостаточность вычислительной мощности
	Риски применения на объекте выработанных в нормальных условиях управленческих решений (категория «П»)	П-1 – слабая синхронизация бизнес-процессов и разработанных решений П-2 – недостаточная квалификация пользователей П-3 – отказ от реализации управленческих решений

Особое внимание при анализе рисков следует уделить так называемым системным рискам (категория «С» в табл. 3.1) как возможным рискам нарушения функционала либо появления системных дисфункций ИС. Как правило, такие риски наиболее критичны, так как приводят к частичной либо полной потере эффективности и результативности ИС. Описание рисков и их последствий свести в таблицу 3.2.

Таблица 3.2 – Риски и последствия рисков

Шифр	Риск	Последствия риска

Качественное оценивание рисков

В подразделе 2 необходимо кратко описать процедуру оценивания рисков. Для оценивания по обеим шкалам используется трехбалльная ранговая шкала, отраженная на рис. 3.1. Результаты оценивания представить в таблицах 3.3 и 3.4.

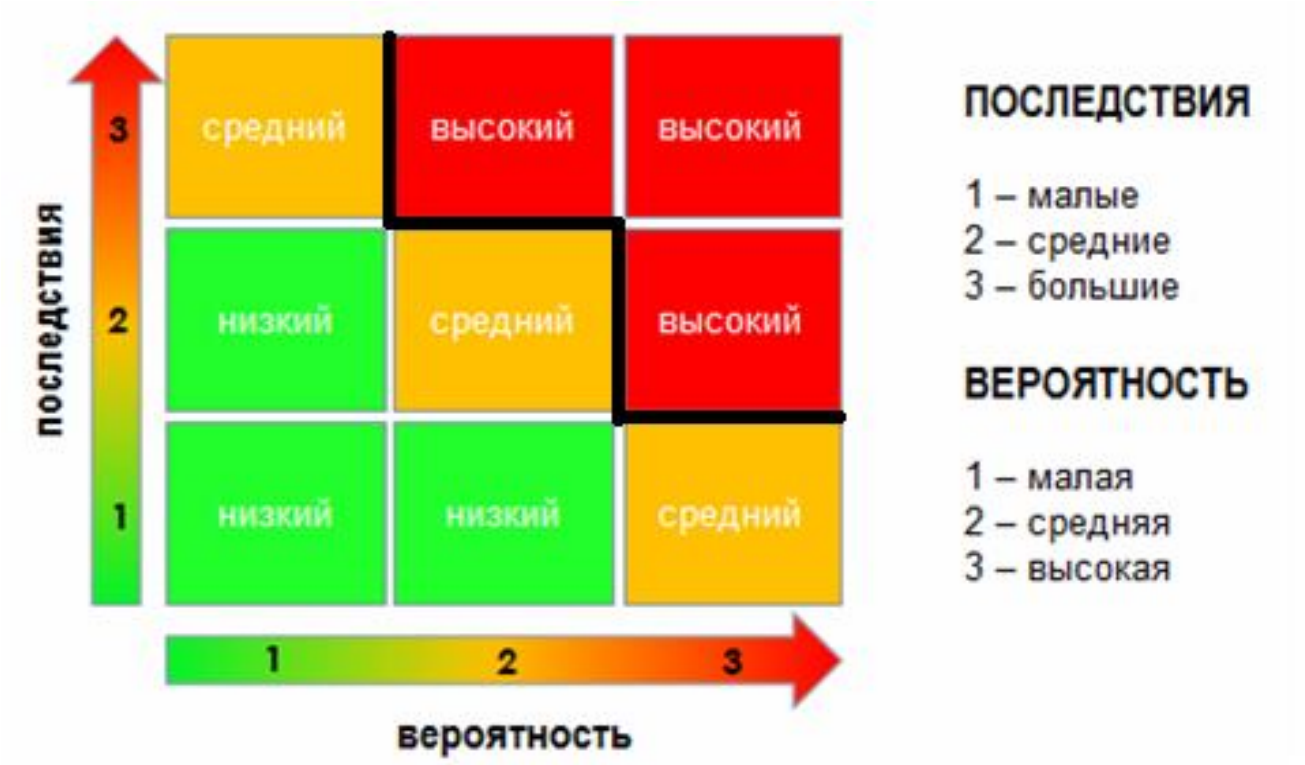


Рисунок 3.1 – Частный пример карты рисков

В табл. 3.3 диапазон вероятности реализации определить априорно.

Таблица 3.3 – Шкала оценивания вероятности реализации риска

Шифр	Вероятность реализации	Диапазон вероятности реализации

В табл. 3.4 существенность последствий кратко описать в пределах 10-20 слов для каждого риска.

Таблица 3.4 – Шкала оценивания существенности последствий риска

Шифр	Оценка существенности	Существенность последствий

В подразделе 3 необходимо отразить процедуру построения карты рисков. Для удобства составления карты рисков необходимо вынести в таблицу 3.5 выявленные риски в соотнесении по шкалам оценивания существенности последствий и вероятности наступления риска. Ранжирование проводить сначала по вероятности, а затем по существенности рисков.

Таблица 3.5 – Соотнесение риска с вероятностью реализации и существенностью последствий

Шифр	Вероятность реализации	Оценка существенности

Частный пример карты рисков представлен на рис. 3.1. На карте рисков вероятность (или частота) отображается по горизонтальной оси, а существенность последствий по вертикальной оси. В этом случае вероятность появления риска увеличивается слева направо по горизонтальной оси, а существенность последствий риска увеличивается сверху вниз по направлению вертикальной оси.

Красным цветом на карте выделены области критических рисков: риски, у которых высокая вероятность появления и тяжелая существенность последствий свершения риска. На эти риски управленцу необходимо обратить внимание в первую очередь: либо смягчить последствия, либо вообще исключить свершение этих рисков.

Черная жирная линия над областью допустимых рисков – это граница толерантности или критическая граница терпимости к риску. Риски, лежащие на границе толерантности, необходимо смещать в область управляемых рисков, чтобы они не перешли в область критических рисков.

Оранжевым цветом выделены риски границы терпимости, у которых средняя частота появления и терпимые последствия свершения рискового события. Эти риски решаются управленцем во вторую очередь, так как у них терпимые последствия и по сравнению с критическими рисками они могут подождать. Однако не следует забывать, что эти риски могут легко перейти в красную область критических рисков.

Зеленым цветом выделены риски, которые редко свершаются и имеют незначительные последствия. С этими рисками работают в последнюю очередь, так как они не несут больших потерь и являются управляемыми в рабочем порядке.

Воздействие на риски

В подразделе 4 на основании полученного материала исследования рискового фона необходимо сделать заключение о подготовке и реализации необходимых превентивных мероприятий, достаточных для приведения объекта информатизации к состоянию умеренного рискового фона, находящегося на карте рисков левее линии толерантности. Заключение должно быть развернутым и содержать:

- вывод о целесообразности применения ИС по назначению;
- прогноз изменения карты рисков в перспективе;

- решения по возможным мероприятиям, оптимизирующим рисковый фон на объекте управления.

Выделенные мероприятия целесообразно представить в виде таблицы 3.6. Последствия риска могут быть продублированы из табл. 3.2 или быть модифицированы с учетом проведенного исследования рискового фона при выполнении пунктов задания 2 и 3.

Таблица 3.6 – Последствия рисков и мероприятия для оптимизации рискового фона объекта управления

Шифр риска	Последствия риска	Мероприятия по купированию последствий риска

Следует отметить, что реализация качественного и высоко эффективного риск-менеджмента – задача сложная и требует применения нетривиального математического аппарата и построения математической модели мониторинга и регулирования рискового фона объекта управления.

Контрольные вопросы и упражнения

1. Что такое карта рисков?
2. Какие методы ранжирования существуют?
3. Что такое шкала существенности карты рисков?
4. Что такое шкала оценивания вероятности возникновения риска?
5. На какие риски стоит обращать внимание в первую очередь?
6. Что такое критическая граница терпимости риска?