УДК 004.4:006.86

МЕТОД ПОВЫШЕНИЯ КАЧЕСТВА РАЗРАБОТКИ ПРОГРАММНОГО ОБЕСПЕЧЕНИЯ С ИСПОЛЬЗОВАНИЕМ МЕТОДОЛОГИИ TETRIS

Гаранин И.В., Садковская Н.Е.

МИРЭА – Российский технологический университет, Москва,
e-mail: Garanin_i.v@mail.ru, natsadkovskaya@rambler.ru

Для повышения качества программного обеспечения можно сосредоточиться на качестве самого продукта, делая его более понятным и доступным пользователю. Качество начинается с удовлетворения потребностей разработчиков, точного соблюдения процессов и удачных технологических подходов. Без полного понимания того, что, зачем и как делается, ни одна строка в программе не должна быть изменена. Достижение качества – обязанность каждого разработчика. Качество доказывается удовлетворением потребностей пользователей. В настоящий момент при разработке программного обеспечения применяют две основные методологии: «Гибкая методология – Agile», а также последовательная методология «Водопад – Waterfall». В данной статье представлен опыт внедрения программного продукта Tetris. Внедрение Tetris сопряжено с решением проблем, которые были определены на старте проекта во всех методологиях. Определены численные показатели работы до и после внедрения данного метода. Выпуск программного обеспечения в наши дни нельзя представить без применения самых новейших методов управления и бизнес-инжиниринга. В данной статье проанализированы и определены наиболее распространенные проблемы, возникающие в процессе разработки, приведены основные методы повышения его качества при помощи метрологии «Tetris», показан пример ее внедрения.

Ключевые слова: разработка программного обеспечения, менеджмент качества, стандартизация, обеспечение качества, Kanban, Scrum, Waterfall, Tetris

METHOD FOR IMPROVING THE QUALITY OF SOFTWARE DEVELOPMENT USING THE TETRIS METHODOLOGY

Garanin I.V., Sadovskaya N.E.

MIREA – Russian technological university, Moscow,
e-mail: Garanin_i.v@mail.ru, natsadkovskaya@rambler.ru

To improve the quality of software, you can focus on the quality of the product itself, making it more understandable and accessible to the user. Quality starts with meeting the needs of developers, strict adherence to processes and successful technological approaches. Without a full understanding of what, why and how is done, not a single line in the program should be changed. Achieving quality is the responsibility of every developer. Quality is proven by user satisfaction. At the moment, two main methodologies are used in software development: «Agile methodology – Agile», as well as a consistent methodology «Waterfall – Waterfall». This article presents the experience of implementing the Tetris software product. The introduction of Tetris involves solving problems that were identified at the start of the project in all methodologies. The numerical indicators of work before and after the introduction of this method are determined. The release of software today is unthinkable without the use of the latest methods of management and business engineering. This article analyzes and identifies the most common problems that arise during the development process. The main methods of improving its quality with the help of metrology Tetris are given, an example of its implementation is given.

Keywords: software development, quality management, standardization, quality assurance, flexible software development methodologies, Kanban, Scrum, Waterfall, Tetris

Разработка программного обеспечения в рамках организации осуществляется командами. Перечислим состав типичной команды, работающей по гибкой методологии Agile. В ее состав входят руководитель продукта (отвечает за конечный результат продукта), дизайнер интерфейса, системный аналитик, разработчик и тестировщик. Нужно отметить, что команды, работающие по методологии «Waterfall», имеют точно такой же состав, что и работающие по гибкой методологии «Agile», за исключением руководителя продукта, на его месте стоит руководитель ИТ-проекта.

В последнее время ввиду возросшей скорости и, как следствие, высокой нагрузки на членов команд обострились вопросы планирования сроков, качества и прозрачности процессов разработки программного обеспечения.

Повышение качества программного обеспечения регулируется международными стандартами ISO/IEC 25010:2011, ISO 9001:2015.

Для улучшения качества процесса разработки программного обеспечения было принято решение о применении метода «Tetris». Tetris развивается с учетом требований команд и автоматизирует в себе все больше ИТ-процессов. Рассмотрим его внедрение в отдельно взятой крупной российской ИТ-компании. В данной ИТ-компании

четко прослеживается использование 3 фреймворков:

• Kanban (относится к семейству гибкой методологии – Agile);

• Scrum (относится к семейству гибкой методологии – Agile);

• Waterfall (относится к последовательной методологии – «Водопад») [1, с. 127].

Подавляющее большинство команд, вовлеченных в проекты, работает по Scrum – около 90%. Tetris на данный момент базируется именно на Scrum, что позволяет его безболезненно интегрировать в работу команд. Tetris не только использует принципы и ценности Scrum, но и привносит свои.

Департамент ИТ имеет в своей структуре множество команд разработки, каждая из которых либо реализует часть функционала, либо полностью разрабатывает программный продукт. До внедрения Tetris руководство использовало классические метрики, присущие современным бизнес-моделям, такие как KPI (оценка по ключевым параметрам эффективности команд) и SLA (соглашение об уровне сервиса). Также использовались метрики эффективности разработки, такие как Time to Market (время вывода ПО в промышленную эксплуатацию) и диаграмма «сгорания» ресурсов (инструмент для наблюдения прогресса по проекту и контроля выполнения плана). Следует отметить, что данные метрики не позволяли видеть реальное состояние дел в командах, а лишь давали менеджменту ИТ текущее состояние, соответственно, внесение коррективов в управление командами осуществлялось с задержкой, исправлялась не сама причина проблемы, а только следствие процессов, происходящих внутри команд.

В результате процесса внедрения был проведен анализ проблем и выявлено, что внедрение программного продукта Tetris позволяет:

• автоматизировать планирование реализации крупных ИТ-задач/проектов;

• повысить скорость и качество такого экспресс-планирования;

• выработать постоянный и комфортный темп работы за счет более качественного планирования задач.

Для компании в целом Tetris позволяет:

• автоматизировать сбор метрик эффективности продуктовых команд и ИТ-процессов для последующей их оптимизации;

• получить на более раннем сроке информацию о плановых сроках реализации задач и необходимых ресурсах для последующего формирования управленческих решений;

• реализовать единую, понятную базу, которая позволяет делать процессы прозрачными, а коммуникацию между членами команды – более эффективной.

В начале процесса по внедрению Tetris было принято решение внедрять его в командах, которые использовали в своей работе методологию Scrum.

Материалы и методы исследования

На текущей стадии Tetris может использоваться в командах, работающих по гибким методологиям Scrum и Kanban. Команды знают основные принципы работы с гибкой методологией и успешно по ней работают. При внедрении в команды программного продукта Tetris есть обязательные события, без которых основные принципы Tetris могут быть не реализованы. Это обусловлено тем, что результаты деятельности команды, работающей по гибкой методологии, используются, в том числе, в процессах автоматического прогнозирования реализации задач.

Примером могут служить:

• оценка задач в условных единицах сложности реализации (где 1 – легко, а 5 – очень сложно);

• декомпозиция задач (разбиение большой задачи на подзадачи);

• ретроспектива при реализации подзадач (оценка уже сделанной работы).

Существуют 3 вида политики установки изменений на продуктивные среды:

• релизная политика – это когда все изменения собираются в один релиз, который в свою очередь устанавливается на продукционные серверы;

• атомарная – каждое изменение может ставиться сразу после его тестирования;

• смешанная – когда есть ряд изменений, которые могут ждать релиза, а другие могут ставиться сразу [2, с. 228].

Tetris поддерживает все виды процессов имплементаций, различия будут только в схеме процессов.

Как и в методологиях Scrum или Kanban, в Tetris есть свои обязательные этапы, от регулярности проведения которых напрямую зависит не только качество работы Scrum-команды, но и алгоритм автоматического Tetris-планирования:

• этап планирования итераций разработки;

• ежедневная встреча членов команд;

• этап обзора итогов определенной итерации разработки;

• ретроспектива определенной итерации разработки [2, с. 229].

На каждом из этих этапов важно выполнять все шаги всеми командами, чтобы

результаты деятельности команд и последующая синхронизация между ними были «прозрачными» и однозначно воспринимались всеми участниками процесса.

Можно привести следующий пример.

Три команды совместно работают над одним проектом. Две команды правильно оценили свои задачи в условных единицах сложности реализации, они используют общие процессы задач и минимум раз в день актуализируют статус своих задач, декомпозируют бизнес-задачи на ИТ-задачи, а третья команда не уделяет должного внимания вышеперечисленным принципам. Данный подход приводит к тому, что невозможно будет:

• посчитать прогресс по интеграционной задаче;

• включить статистические данные по задачам для программного продукта Tetris;

• получить централизованно ретроспективные данные команды для последующего анализа.

Tetris помогает в декомпозиции задач руководителю ИТ-проекта, так как использует две концепции, два базовых подхода к разделению и декомпозиции крупных задач на пользовательские истории – «горизонтальное» и «вертикальное» разбиение. В Tetris используются оба подхода, только на разных уровнях иерархии задач.

Разделение – процесс, при котором одна большая сущность (Epic) разделяется на сущности (Task), которые сами по себе ценны и независимы (Истории, Задачи) по отношению к другим таким же сущностям [3, с. 111].

Декомпозиция – процесс, при котором независимая сущность (История, Задача) декомпозируется на дочерние сущности, которые имеют прямую зависимость от основной независимой сущности. Такие сущности не имеют самостоятельной ценности и всегда подчинены своему родителю. При удалении основной сущности дочерние сущности утрачивают свой смысл.

Оценка задач в Tetris в условных единицах сложности реализации принадлежит к классу относительных оценок. Это значит, что задачи оцениваются друг относительно друга. Берутся те задачи, которые нужно оценить, и ранжируются по возрастанию затрачиваемых на них усилий. Условными единицами измеряют усилия, которые нужны для выполнения элемента задачи продукта или любого другого этапа работы. Пользуясь условными единицами, руководитель ИТ-проекта или руководитель продукта присваивает каждой подзадаче некое количественное значение. Сами по себе эти количественные оценки не важны. Важно то, как оценки разных элементов соотносятся друг с другом.

Например, подзадача со значением S должна быть вдвое больше подзадачи со значением XS и соответствовать двум третям подзадачи со значением M, как показано в таблице 1.

Если команда, работающая по фрэймворку Scrum или Kanban, абсолютно новая, и с задачами, которые лежат в продуктовом списке на реализацию, она сталкивается первый раз, это означает, что опереться на ретроспективные оценки она не может. В данном случае рекомендуется использовать метод «Выстраивания порядка». Суть его проста – выбирается самая простая задача в списке относительно других задач, и ей присваивается значение XS, то же самое осуществляем с самой сложной задачей – присваиваем значение XL. Все остальные задачи средней сложности должны быть классифицированы по остальным значениям сложности: S, M, L. Таким способом у нас будут определены градации сложности всех наших задач [3, с. 112].

Если команда проработала совместно пару проектов, это означает, что у нее уже сформировалась градация сложности задач, с которыми она сталкивается. В данном случае при оценке очередной задачи или подзадачи стоит учитывать оценки, которые были даны в предыдущих периодах, это позволит повышать точность оценки.

Таблица 1

Оценка задач в Tetris

Сложность задачи (от 1 до 10, где 1 – легкая задача, 10 – чрезвычайно сложная)	Сроки реализации в рабочих днях	Значение сложности в условных единицах реализации	Оценка реализации в человеко-часах
1	5	XS	0–32
2	10	S	32–64
3	15	M	64–96
5	25	L	96–160
8	40	XL	160–256

Ежедневные встречи команд называют стендапами (Scrum Stand Up), или просто встречами, собраниями (meeting). В процессе стендапа каждый член команды высказывается по трем темам:

• что было сделано вчера?

• что будет сделано сегодня?

• какие имеются сложности в достижении результатов? [4, с. 56].

Фокус внимания в этих вопросах стоит именно на результате деятельности, а не на процессе. Факт «что-то будет делаться» не гарантирует положительного эффекта.

Для корректной работы Tetris критически важно, чтобы участники команды во время обсуждения актуализировали статусы задач на доске разработки. Обусловливается это тем, что надолго забытые задачи негативным образом сказываются на статистических показателях, которые считываются (в том числе учитываются) в таком показателе, как «Время выполнения задачи».

Как правило, очередная итерация реализации задачи (спринт) завершается ретроспективой.

Ретроспектива – это мероприятие, направленное на улучшение командных процессов за счет обсуждения предыдущих событий и процессов в команде, которые наблюдались в течение спринта. Прямой целью ретроспективы является повышение эффективности процессов внутри команды. В результате ретроспективы команда может принять новые решения о том, как можно работать лучше и результативнее.

В Tetris особую роль играют оценки, которые были даны задачам перед началом спринта. На их основе в автоматическом режиме собирается статистика о трудоемкости задач в разрезе каждой компетенции. Поэтому очень важно обсуждать на ретроспективе точность оценок. Однако после завершения спринта эти оценки могут быть не совсем точными ввиду появления новых рисков или неопределенности, которые повлияли на сложность выполнения задачи.

Поэтому на ретроспективе нужно обращаться к оценкам, которые были даны перед началом спринта, и сопоставлять с фактической сложностью задач.

Точность той или иной условной единицы сложности реализации задачи слагается из следующих критериев оценки:

• трудоемкость;

• риски;

• неопределенность.

К этим трем параметрам можно добавить еще один критерий – продолжительность. Как раз последний критерий отслеживается в автоматическом режиме, например с использованием программного обеспечения мониторинга реализации проекта Atlassian Jira. На рисунке 1 показан пример, где настроен отчет в плагине Structure, который показывает плановые оценки по задачам и фактические сроки реализации задач. С учетом постоянного использования данного отчета для подготовки к ретроспективам можно процесс оценки задач постоянно совершенствовать.

Для того чтобы участникам команд было проще оценивать новые задачи, создается матрица сопоставления референтных задач (рис. 2). Референтные задачи – это типовые задачи, которые хорошо знакомы команде. Также в такую матрицу можно внести информацию по типам задач, чтобы облегчить оценку разных типов задач.

В случае долгосрочного планирования своих задач используется специально разработанный плагин, который позволяет осуществлять экспресс-оценку задач.

Потребность в создании такого плагина возникла из-за отсутствия в настоящее время:

• единого подхода к планированию задач;

• согласованного с бизнесом алгоритма планирования;

• инструментов анализа задач [5, с. 1].

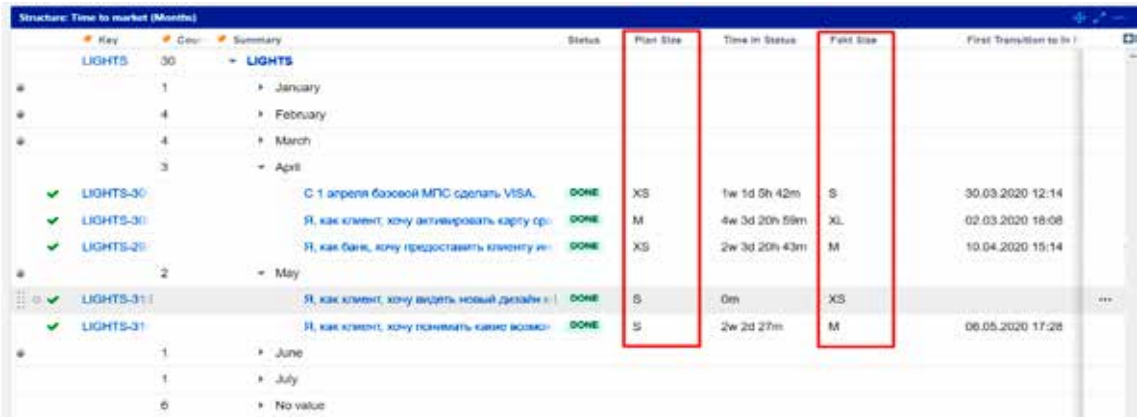

Рис. 1. Пример отчета по ретроспективе спринта

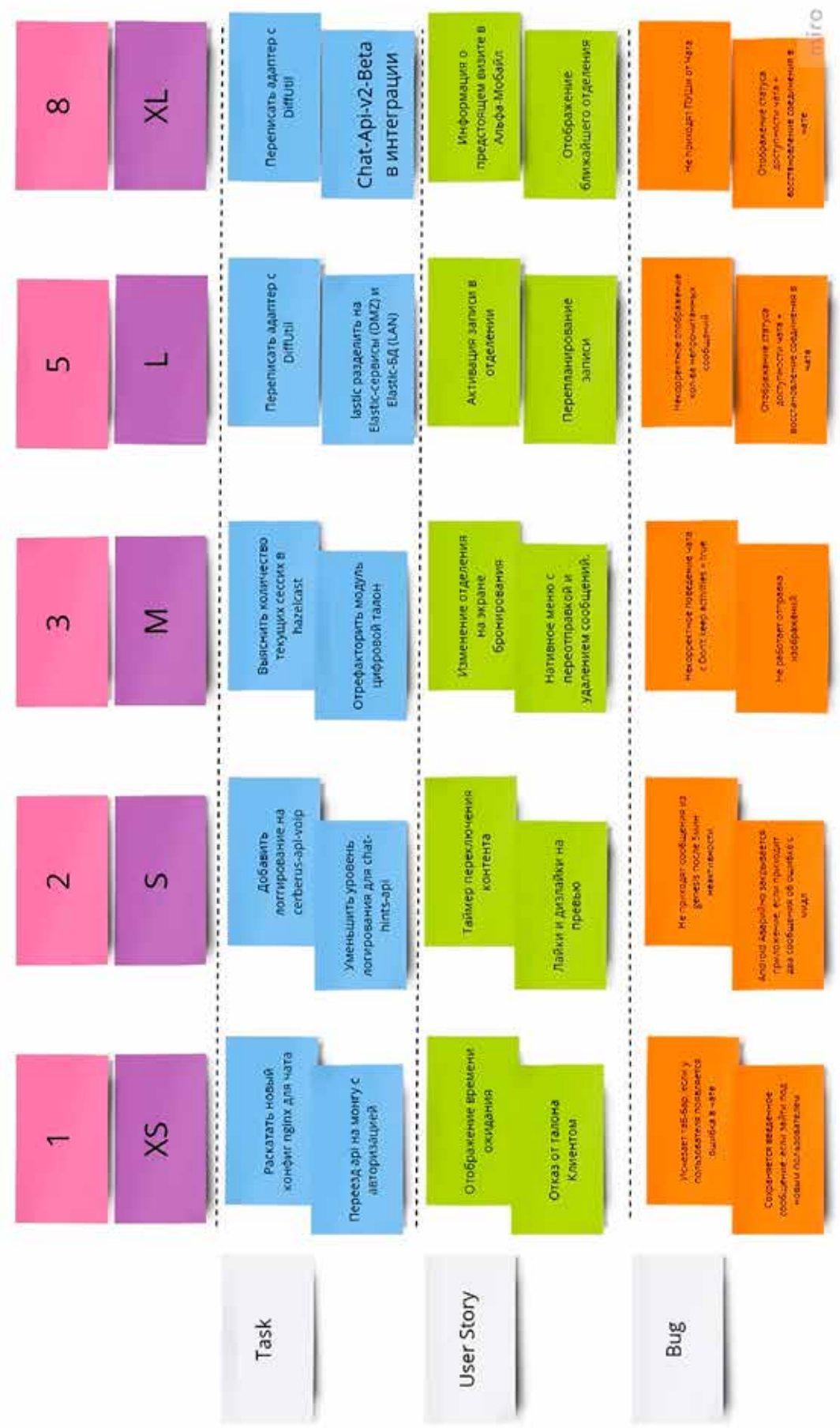

Рис. 2. Пример матрицы задач

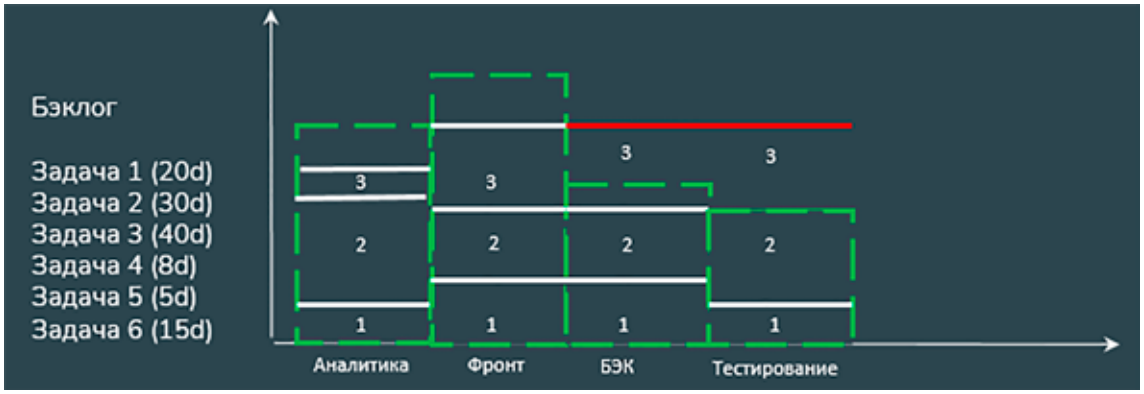

Рис. 3. Визуализация при планировании задач

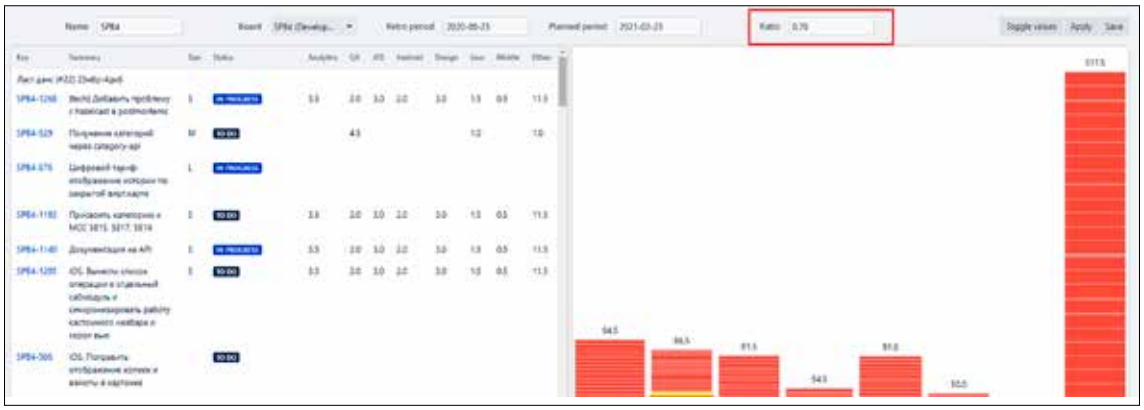

Рис. 4. Пример отчета в системе Jira для Tetris

Таблица 2

Отчет по пропускной способности в Tetris

Компетенция	Пропускная способность за спринт	Задачи	Аналитик	Разработчик iOS/Android	Разработчик Java	Тестировщик
Аналитик	10	Задача 1	2	3	3	4
Разработчик iOS/Android	12	Задача 2	5	5	7	7
Разработчик Java	8	Задача 3	4	5	4	3
Тестировщик	6	Задача 4	4	5	3	4

Автоматизированное планирование в данный момент реализовано в виде плагина к системе планирования проектов, например Atlassian JIRA (рис. 3).

Для расчета планирования исходными данными являются:

- задачи и их типы;
- оценки задач;
- статусы;
- компетенции.

Основная задача при планировании заключается в определении состава задачи, который может быть проведен конкретной командой, на определенном периоде времени и представлен в виде отчета (рис. 4).

В рамках ручного и/или автоматизированного планирования, основанного на Tetris, можно использовать несколько вариантов взаимодействия в зависимости от характера задачи.

Возможны построения отчетов, показывающие общее количество задач на команду и пропускную способность команд выполнять задачи (табл. 2).

Таблица 3

Отчет по внедрению

Метрики успешности	Работа команды с внедренным Tetris	Работа команды без Tetris
Успешность планирования	Количество возвратов функциональных требований на доработку (штук): **0** Отставание от плана (дней): **0**	Количество возвратов функциональных требований на доработку (штук): **5** Отставание от плана (дней): **4**
Успешность внедрения	Насколько выполнена изначальная цель (%): **99** Все ли дефекты были поправлены? **Да** Если нет, какой срок отставания (дней): **0** Пришлось ли срочно останавливать релиз в связи с критическими дефектами? **Нет** Срок установки ПО на 100% пользователей: **в среднем около 1–2 часов**	Насколько выполнена изначальная цель (%): **74** Все ли дефекты были поправлены? **Нет, оставалось в среднем 7–9 дефектов** Если нет, какой срок отставания (дней): **2–3** Пришлось ли срочно останавливать релиз в связи с критическими дефектами? **Да, 1–2 раза за шесть спринтов** Срок установки ПО на 100% пользователей: **в среднем около 4–5 часов**
Количество дефектов	Количество обнаруженных и заведенных дефектов (штук): **5** Количество необнаруженных дефектов: **1 дефект на 6 спринтов**	Количество обнаруженных и заведенных дефектов (штук): **17** Количество необнаруженных дефектов: **в среднем 7–9 дефекта на шесть спринтов**

Результаты исследования и их обсуждение

Для проверки эффективности внедрения Tetris было проведено исследование качества работы команд.

Оценке подвергалась работа одного пилотного ИТ-направления до внедрения Tetris и после его внедрения, в одинаковом временном промежутке, равном шесть спринтов (или три календарных месяца) (табл. 3).

Результаты, представленные в таблице 3, показывают, что внедрение Tetris влияет как на успешность планирования (улучшение показателей до базового заданного значения), так и на успешность внедрения (показатели в среднем улучшились на 50% от всех приведенных метрик) и количество дефектов. Значения получены на однородных проектах, с одинаковыми оценками трудозатрат во избежание искажений результатов по данным метрикам.

Выводы

По результатам исследования внедрения Tetris можно сформулировать следующие выводы.

• Уменьшены риски совершения ошибок при планировании, разработке и тестировании задач ввиду того, что руководители ИТ видят состояние загрузки внутри команд и могут вносить коррективы в процессы разработки.

• Значительно повышено качество планирования задач для команд. Члены команды понимают, что ИТ-руководитель не перегружает их, а осуществляет детальное планирование задач.

• Возросла мотивация членов команды из-за отсутствия фактора перегрузки задачами, которые без переработок невозможно выполнить.

• Сократилось на 40% время, затрачиваемое на проведение ежедневных синхронизационных встреч, ввиду того, что руководство ИТ получило инструмент, позволяющий видеть загрузку команд и прогнозировать, какие задачи и когда можно ставить в очередь на исполнение.

Приведенные данные показывают, что описанный инструмент повышает производительность и удовлетворенность команд разработки, позволяет оперативно мониторить загрузку команд задачами и вносить коррективы в планирование и приоритизацию, и, как следствие, повысить качество создаваемого конечного ПО.

Список литературы

1. Peter Koning Agile Leadership Toolkit: Learning to Thrive with Self-Managing Teams (The Professional Scrum Series). Addison-Wesley Professional. 2019. P. 125-128.

2. Sibert C., Speicher J., Gray W.D. Less is more: Additional information leads to lower performance in TETRIS models, Proceedings of ICCM 2019. 17th International Conference on Cognitive Modeling. 2020. P. 228-234.

3. Simon Reindl, Stephanie Ockerman Mastering Professional Scrum: A Practitioners Guide to Overcoming Challenges and Maximizing the Benefits of Agility (The Professional Scrum Series). Addison-Wesley Professional. 2019. P. 111-114.

4. Collins E., deLucena V. Software test automation practices in agile development environment: An industry experience report. In Proceedings. 7th International Workshop on Automation of Software Test. AST'12. IEEE Computer Society, Washington, DC. 2019. P. 55-63.

5. Agile-манифест разработки программного обеспечения [Электронный ресурс]. URL: https://agilemanifesto.org (дата обращения: 21.03.2022).