DOI: 10.37791/2687-0657-2024-18-6-84-98

Особенности внедрения гибких методологий в процесс разработки программного обеспечения в сфере телекоммуникационных услуг

А.Ю. Анисимов[1*], А.Е. Трубин[1], Е.Н. Токмакова[1], Е.В. Филимонова[1]

[1]*Университет «Синергия», Москва, Россия*

[*]*anisimov_au@mail.ru*

Аннотация. В статье рассматриваются особенности внедрения гибких методологий в процесс разработки программного обеспечения в сфере телекоммуникационных услуг. Большая потребность в ИТ-продуктах заставляет их производителей искать новые решения и подходы, способствующие сокращению времени на разработку без негативного влияния на качество конечного продукта. Изменение процессов разработки ПО затрагивает как его техническую, так и бизнес-сторону, в том числе управление. Несмотря на существование большого количества современных методологий управления, выбор наиболее подходящей является сложным процессом, так как требует тщательной постановки целей и критериев поиска. От того, насколько правильно подобрана модель управления разработкой ПО, зависит успех как компании-исполнителя, так и компании-заказчика. Целью исследования является унификация гибкой методологии управления разработкой ПО в сфере телекоммуникационных услуг. Задачами являются модернизация системы управления разработкой ПО на базе Kanban и языка UML и проработка этапов внедрения методологии. Методическую основу исследования составили такие методы, как анализ и систематизация, которые включают системный, комплексный и сравнительный подходы, а также эксперимент. Научная новизна работы заключается в создании унифицированной модели управления разработкой ПО для телекоммуникационного сектора. В статье отмечается, что основу унифицированной модели составляют этапы внедрения методологии Kanban, посредством которой было сокращено количество фаз процесса управления разработкой ПО и, следовательно, время на реализацию проекта. Авторы подчеркивают, что разработанная унифицированная модель может быть использована не только в управлении разработкой ПО сферы телекоммуникационных услуг, но и в любом другом проекте разработки ПО, независимо от сферы деятельности.

Ключевые слова: управление разработкой ПО, гибкие методологии, Agile, Kanban, UML

Для цитирования: *Анисимов А.Ю., Трубин А.Е., Токмакова Е.Н., Филимонова Е.В.* Особенности внедрения гибких методологий в процесс разработки программного обеспечения в сфере телекоммуникационных услуг // Современная конкуренция. 2024. Т. 18. № 6. С. 84–98. DOI: 10.37791/2687-0657-2024-18-6-84-98

Features of Introducing Agile Methodologies in the Software Development Process for Telecommunications Services

A. Anisimov[1*], A. Trubin[1], E. Tokmakova[1], E. Filimonova[1]

[1]*Synergy University, Moscow, Russia*

[*]*anisimov_au@mail.ru*

Abstract. The article discusses the features of the introduction of agile methodologies in the process of software development (hereinafter referred to as software) in the field of telecommunications services. The great need for IT products forces their manufacturers to look for new solutions and approaches that help reduce development time without negatively affecting the quality of the final product. Changing software development processes affects both its technical and business side, including management. Despite the existence of a large number of modern management methodologies, choosing the most appropriate one is a difficult process, as it requires careful setting of goals and search criteria. The success of both the implementing company and the customer company depends on how well the software development management model is selected. The purpose of the study is to unify agile methodology for managing software development in the field of telecommunications services. The objectives are to modernize the software development management system based on Kanban and the UML language and to work out the stages of implementing the methodology. The methodological basis of the research was based on such methods as analysis and systematization, which include systematic, complex and comparative approaches, as well as experiment. The scientific novelty of the work lies in the creation of a unified software development management model for the telecommunications sector. The article notes that the basis of the unified model is the stages of implementation of the Kanban methodology, through which the number of phases of the software development management process and, consequently, the time for project implementation was reduced. The authors emphasize that the developed unified model can be used not only in the management of software development in the field of telecommunications services, but also in any other software development project, regardless of the field of activity.

Keywords: software development management, Agile methodologies, Agile, Kanban, UML

For citation: Anisimov A., Trubin A., Tokmakova E., Filimonova E. Features of Introducing Agile Methodologies in the Software Development Process for Telecommunications Services. *Sovremennaya konkurentsiya*=Journal of Modern Competition, 2024, vol.18, no.6, pp.84-98 (in Russian). DOI: 10.37791/2687-0657-2024-18-6-84-98

Введение

В настоящее время разработка программного обеспечения (ПО) является одним из самых важных видов деятельности. Практически в каждой отрасли используется специальное программное обеспечение, которое является основой функционирования всего предприятия. Именно поэтому эффективно построенный процесс разработки ПО – это основополагающая часть успешного ИТ-проекта.

Актуальность обусловлена комплексностью процесса разработки ПО и, как следствие, необходимостью осмысления приме-

нения методик гибкого контроля проектами на всех этапах разработки [5].

Одной из быстро развивающихся отраслей в сфере информационных технологий является сфера разработки телекоммуникационных приложений, так как в современном мире сложно представить жизнь без телевидения, интернета или мобильной связи.

С целью предоставления качественных телекоммуникационных услуг компании необходимо не только качественное оборудование, правильно подобранный персонал, а также внедрение постоянных новшеств с целью противостояния конкуренции на рынке, но и разработка ПО, позволяющего обеспечить эффективное функционирование всех предоставляемых услуг связи.

Для того чтобы удовлетворить потребности конечного пользователя, компания должна следовать современным тенденциям на рынке телекоммуникационных технологий и предлагать клиентам современные услуги интернет- и телефонной связи. Кроме того, выпуск ПО должен осуществляться стабильно и в четко обозначенные сроки. Для достижения поставленных целей требуется эффективно налаженный процесс разработки ПО.

Согласно динамике перераспределения CAPEX, большую часть капитальных затрат в телекоммуникационном секторе составляют траты на программное обеспечение. Это значит, что в последние десятилетия наблюдается значительное смещение акцентов в сторону телекоммуникационного программного обеспечения. Компании стали меньше тратить бюджет на закупку нового оборудования. Улучшение качества связи обеспечивается за счет внедрения нового современного ПО в архитектуру телекоммуникационных систем.

Цели и задачи исследования

Целью исследования является унификация гибкой методологии управления разработкой ПО в сфере телекоммуникационных услуг. Задачами являются модернизация системы управления разработкой ПО на базе Kanban и языка UML и проработка этапов внедрения методологии.

Методы исследования

Методическую основу исследования сформировали следующие методы: анализ и систематизация, которые включают системный, комплексный и сравнительный подходы, а также эксперимент.

Анализ существующих методологий для управления разработкой ПО

Появление гибких методологий в разработке ПО полностью изменило подход к реализации данного процесса в целом. Процессы стали более адаптивными к изменениям, улучшилась коммуникация между участниками проекта, увеличилась концентрация на продукте, между заказчиком и исполнителем стали выстраиваться партнерские отношения [10, 13]. На сегодняшний день трудно представить разработку ПО без использования гибких методологий.

На данный момент существует множество гибких методологий управления разработкой ПО, за основу которых взяты принципы Agile [4]. Детальный сравнительный анализ различных методологий был произведен авторами данной статьи ранее [2].

Одним из самых известных подходов для компаний телекоммуникационной сферы является Scrum. Скрам-команда включает в себя всех специалистов, необходимых для выполнения задачи. Чаще всего команда состоит из 5–7 участников. Управление и организация работы внутри команды осуществляется скрам-мастером (Scrum Master). Формированием требований и приоритезацией в скрам-команде занимается владелец продукта (Product Owner).

Работа в команде выполняется фиксированными итерациями – спринтами. В конце каждого спринта команда должна предоставить готовую часть функционала, которую можно отдать заказчику. Каждый день команда проводит митинги, на которых члены команды синхронизируют свою работу и определяют план работы на предстоящий день. В конце каждого спринта проводится инспекция (обзор спринта), на котором команда получает обратную связь от заказчика. Кроме того, команда проводит ретроспективы, необходимые для улучшения процессов внутри команды [14].

Среди наиболее известных гибких методик выделяют также Kanban. Если рассматривать его с точки зрения использования в ИТ, то следует отметить, что в разработке ПО Kanban больше рассматривают как инструмент визуального менеджмента, в котором выделяют следующие принципы [8]:

1) визуализация прогресса;
2) ограничение объема незавершенной работы;
3) фокусирование на процессе;
4) совершенствование процесса;

Основным инструментом в Kanban является доска, на которой размещаются задачи, отсортированные в зависимости от этапа их выполнения. Чаще всего доска создается с помощью специальных программных приложений, например Jira. Но многие команды также практикуют использование обычной магнитной доски в качестве инструмента визуализации ежедневных задач [3].

Как и в Scrum, в Kanban есть ежедневные встречи, на которых команда обсуждает текущий статус работы и ставит цели на предстоящий день. Но, как отмечает Е.П. Зараменских, Kanban нельзя использовать в качестве инструмента управления жизненным циклом ПО. Его необходимо применять в совокупности с другими методологиями, так как Kanban не предлагает самостоятельных методик и практик [8].

eXtreme Programming (XP) – экстремальное программирование – это еще одна методика разработки, построенная на принципах Agile. Данную методику используют в основном в проектах разработки с нестабильными требованиями и недокументированными процедурами [11]. Основным принципом в экстремальном программировании является постоянное взаимодействие с заказчиком через его представителя. Во время разработки чаще всего выбираются простые методы, так как, согласно философии экстремального программирования, проще вносить изменения в простую программу, чем пересматривать архитектуру усложненной. Кроме того, код является доступным для всех участников-разработчиков. Отсюда выделяется еще один принцип экстремального программирования – это коллективное владение кодом.

Одной из особенностей, присущих экстремальному подходу в программировании, можно назвать попарное программирование [1]. Использование данного метода подразумевает, что два разработчика, используя одну и ту же машину, осуществляют написание кода, его доработку или реализацию.

Методика экстремальной разработки решает проблему контроля работ благодаря высокому уровню коммуникации внутри команды [1, 6]. Но в то же время в команде, использующей экстремальную методику программирования, высок риск недопонимания между участниками. Кроме того, применение XP будет эффективным только в том случае, когда в команде количество разработчиков пять и более человек. Фокус данной методики сосредоточен на процессе разработки, что, в свою очередь, означает отсутствие в XP-методике практик менеджмента. В данном случае контроль за качеством разрабатываемого ПО должен взять на себя заказчик.

Еще одной гибкой методикой разработки является Feature Driven Development (FDD). Ее основное преимущество состоит в том,

что данная методология позволяет работать большим по численности командам. FDD разделяет работу над проектом на пять процессов:

1. Разработка общей модели.
2. Составление списка необходимых функций системы.
3. Планирование работы над каждой функцией.
4. Проектирование функций.
5. Реализация функций.

Выполнение первых трех стадий осуществляется во время нулевой итерации. Во время процесса реализации уделяется особое внимание проектированию модели системы при помощи цветных UML-диаграмм. Каждая команда предлагает видение своей модели системы. На основе предложенных вариантов разрабатывается модель, которая впоследствии будет взята в качестве основы для разработки. После этого каждая часть модели детализируется и уточняется.

Управление процессом в такой команде осуществляется при помощи ответственных лиц, способных настроить коммуникацию среди команд-участников. Таким образом, даже географически распределенные команды могут работать по методике FDD. Тем не менее методология FDD имеет ряд недостатков, одним из которых является отсутствие коммуникации между разработчиками, так как, по сути, каждая команда работает изолированно [7]. Данный факт противоречит одному из Agile-принципов, который гласит, что «непосредственное общение является наиболее практичным и эффективным способом обмена информацией как с самой командой, так и внутри команды» [1].

Другим недостатком данной методики является отсутствие правил работы с требованиями и документацией. Это означает, что данная методика требует внедрения дополнительной методологии, способной урегулировать правила работы с документацией.

Еще одной методологией, связанной с построением UML-диаграмм, является Model-Driven Architecture and Development (MDA/MDD). Замысел ее заключается в том, что разработка ПО будет реализовываться на базе визуальной архитектурной модели системы. Все изменения, вносимые впоследствии в код программы, будут завязаны на изменениях в визуальной модели [1]. Можно сказать, что основным механизмом в MDA является построение UML-диаграмм.

Процесс разработки приложения с применением MDD-методологии подразумевает составление верхнеуровневой модели приложения на начальных этапах разработки. Эта модель обладает высоким уровнем абстракции и включает в себя основные требования к системе.

Верхнеуровневая модель становится основой для платформенно-независимой модели, которая будет отображать основную семантику приложения. Наконец, последним этапом моделирования становится создание платформенно-зависимой модели, которая и будет использована в качестве основы при написании кода программы.

Наследником MDA-методики является такой подход в разработке, как Domain Driven Development (DDD). В основу этого подхода также заложен принцип разработки ПО на основе его модели. Отличие DDD от MDD заключается в том, что в DDD особое внимание уделяется реализации кода именно с точки зрения бизнес-области разрабатываемого приложения.

Модель DDD больше ориентирована на построение процессов внутри команды разработчиков. Ее основным преимуществом с точки зрения разработки является ограниченный контекст. Это значит, что доменный подход в разработке позволяет разработчикам вносить изменения в код программы, не опасаясь, что эти изменения негативно повлияют на другие модули. С точки зрения управления процессом DDD-подход дает возможность руководителю проекта

в значительной степени распределить или, иными словами, распараллелить задачи, тем самым ускорив процесс разработки.

Подход Domain Driven Development, несмотря на то, что относится к гибким методологиям, является довольной старой моделью процесса управления разработкой ПО. Использование DDD-подхода, по мнению некоторых исследователей, является вполне разумным и эффективным в случае, если в проекте достаточно сложная бизнес-логика, включающая десятки кейсов [9, 12]. Тем не менее этот подход не настолько популярен среди гибких методологий разработки.

Появление гибких методологий обусловлено изменениями в общественных потребностях, которые, в свою очередь, являлись следствием технологического развития. Для того чтобы успешно выполнить проект разработки ПО, необходимы специальные подходы, направленные на улучшение контроля за процессом и качеством создаваемого продукта [10].

Особенности внедрения гибких методологий в процесс разработки ПО в сфере телекоммуникационных услуг

В настоящее время большое внимание уделяется разработке телекоммуникационных сервисов и совершенствованию их технологий. Телекоммуникационные компании пытаются обеспечить высокий уровень качества предоставляемой связи путем внедрения современного высокотехнологичного программного обеспечения. Использование такого подхода позволяет решить основные проблемы, стоящие перед компаниями, а именно повышение качества связи без замены старого оборудования на новое, а также увеличение спектра предлагаемых услуг с целью поддержания конкуренции на рынке.

Kanban является универсальным инструментом для построения процессов в любой сфере деятельности, в связи с чем именно этот инструмент был выбран в качестве основы для создания универсальной модели управления разработкой ПО [3]. Данная методология позволяет решить задачи оптимизации и регулирования работы внутри команды.

Использование инструмента Kanban делает процесс разработки ПО более эффективным, визуализируя его состояние на момент поступления задачи в тестирование.

Исследование эффективности использования инструмента Kanban осуществлялось на основе эксперимента в одной из команд по разработке телекоммуникационных систем компании ООО «ДОЙЧЕ ТЕЛЕКОМ АЙТИ РУС». Выбранная команда занималась построением унаследованной системы управления связью. Исходными данными являлось отсутствие инструмента, позволяющего улучшить и визуализировать процесс работы в команде.

Команда состояла из 7 человек: 1 владелец продукта, 1 скрам-мастер, 3 разработчика и 2 тестировщика. Команда занималась full-stack разработкой как бэкенд, так и фронтенд части приложения. В качестве метода управления разработкой была выбрана методология Scrum.

Исходя из того, что на каждую стадию в каскадной модели разработки выделялось от одного до двух месяцев, процесс разработки в команде разделялся на спринты. Один спринт длился 2 недели, т.е. в среднем за один цикл разработки команда проходила 2–4 спринта. Список задач, которые необходимо выполнить во время спринта, поступал от владельца продукта. На протяжении всего цикла разработки владелец продукта постоянно находился на связи с командой, тем самым контролируя ход работы и, при необходимости, помогая команде разобраться с бизнес-процессами.

Основной проблемой в команде было отсутствие этапа планирования, на котором команда могла бы ознакомиться с предсто-

ящими задачами и оценить время, необходимое для их реализации. Владелец продукта совместно с руководителем команды декомпозировали задачи и распределяли их по предстоящим спринтам. После этого готовый список задач отправлялся на ознакомление команде. Распределение задач проходило на ежедневных встречах (рис. 1).

Узким местом в команде являлась фаза ознакомления и распределения задач. На этапе ознакомления команда имела возможность только изучить основные аспекты предстоящей работы, возможно, уточнить детали, но у участников команды не было возможности оценить время работы, которое необходимо для решения. Как следствие, в процессе разработки увеличивался риск невыполнения той или иной задачи до момента ее выпуска.

Следующим моментом, нуждающимся в улучшении, являлся процесс распределения задач. Руководитель команды распределял задачи из списка между участниками команды на ежедневных встречах. При таком построении процесса выделялись следующие недостатки:

1. Состояние готовности продукта не выглядело прозрачным, как и сам процесс разработки. Единственным человеком, имеющим представление о ходе работы, был руководитель команды.
2. Участники команды неэффективно тратили свое время, анализируя полный список предстоящих задач, так как распределение задач проходило позже.
3. В команде был низкий уровень мотивации по причине отсутствия видения процесса работы.

В таких условиях резко возникла потребность в использовании инструмента, способного решить имеющиеся проблемы в процессе работы. Именно Kanban, как инструмент для визуализации процесса работы команды, позволил избавиться от недостатков в текущем процессе управления.

Стартовой точкой внедрения Kanban в команде было определение ролей в команде. А для эффективной работы Kanban

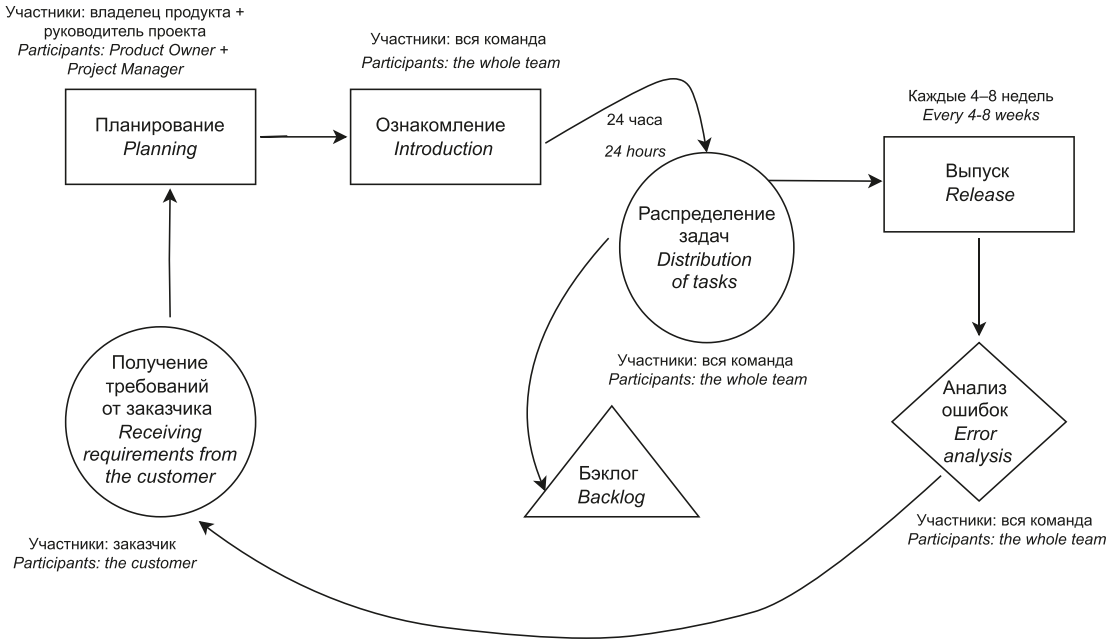

Рис. 1. Классическая модель разработки ПО в команде

Fig. 1. The classic model of software development in a team

в команде необходимы как минимум две направляющие роли.

Первая роль – это менеджер сервиса поставки (Service Delivery Manager). В его задачи входит недопущение препятствий или их ликвидация в потоке поставки. Менеджер сервиса поставки должен не только хорошо знать бизнес-составляющую продукта, но и обладать умением налаживать процессы коммуникации внутри команды. Это значит, что в его обязанности входит наблюдение за процессом работы участников команды и создание атмосферы, позволяющей участникам комфортно работать над выполнением задач.

Вторая роль – это менеджер сервиса запросов (Service Request Manager). Задачи, выполняемые этой ролью, включают в себя управление потоком запросов от заказчика. Иными словами, менеджер сервиса запросов – это человек, находящийся в постоянном контакте с заказчиком. Именно он получает список требований и формирует на их основе задачи, которые должны быть выполнены участниками команды. Менеджер сервиса запросов – это тот человек, который очень хорошо знаком с бизнес-стороной продукта, поэтому в его обязанности также входит помочь команде разобраться с логикой работы продукта.

Следующим этапом внедрения Kanban в управлении разработкой ПО было изменение целей и процесса проведения ежедневных командных встреч. Теперь целью ежедневных встреч являлось не только распределение задач из списка, но и обсуждение статуса текущих задач, а также проблем, возникших во время их реализации.

Еще одним немаловажным этапом при внедрении Kanban являлся выбор инструмента управления разработкой ИС. Таковой инструмент полностью отсутствовал в команде. Список задач хранился в Excel-документе, доступном всем участникам команды.

В качестве инструмента управления процессом разработки был выбран сервис Jira (на практике возможно применение другого аналогичного ПО), так как он уже был внедрен в смежных командах, занимающихся разработкой микросервисных систем. Преимущество использования инструмента Jira заключалось в том, что компании не нужно было оплачивать услуги новой SaaS-системы, а можно было использовать уже купленную ранее подписку, при необходимости увеличив количество пользователей.

Заключительным этапом внедрения Kanban инструмента как средства управления процессом разработки было формирование виртуальной Kanban-доски и создание в системе нового проекта разработки ПО. Так как Jira уже была внедрена в других командах, руководителю команды было необходимо лишь запросить доступы в систему для всех участников команды. Помимо этого, системным администратором по запросу руководителя команды был создан новый проект разработки, за которым должны были закрепляться все задачи, созданные участниками команды.

Первичное внедрение методики Kanban было запланировано с наступлением нового цикла разработки. За две недели до его начала владелец продукта перенес все задачи, запланированные на будущий цикл, из общего Excel-документа в Jira-сервис.

Отдельное внимание было уделено существующим ошибкам в работе программы. Для документации ошибок и описания тестовых сценариев в команде использовалось специальное внутреннее приложение, доступное только участникам команды. В этом приложении тестировщики создавали тестовые сценарии на запланированные тесты, а также описывали найденные ошибки и прикрепляли их к соответствующим тестам. С появлением Jira было решено отказаться от использования этой программы, которая являлась уже устаревшей

и недостаточно удобной в использовании. Все найденные ошибки и запланированные тестовые сценарии должны были документироваться в Jira.

Перед началом нового цикла команда была полностью готова к переходу на процесс работы с использованием методики Kanban. На первой встрече по планированию спринтов вся команда распределила задачи по соответствующим спринтам. Предварительно владелец продукта определил задачи, которые необходимо выполнить в течение цикла, оставшееся количество стори-поинтов было заполнено задачами из технического долга.

После формирования спринтов руководитель команды объявлял о старте спринта, и в Jira автоматически сформировалась Kanban-доска.

Каждый участник команды, как только приступал к выполнению задачи, указывал себя в поле *Assignee* и переводил задачу в статус *In Progress.* После завершения разработки задача переводилась в статус *Code Review* до момента, пока код программы пройдет проверку.

После проверки задача поступала в тестирование и ее статус менялся на *Ready for Test*, а в поле *Assignee* указывался инженер по тестированию. После завершения тестирования инженер по тестированию снова переводил задачу на разработчика. Как только исходный код вливался в основную ветку разработки, статус задачи менялся на *Closed* и она переходила в колонку *Done*. Задачи, которые не успели пройти полный цикл разработки до момента их релиза, снова попадали в *Backlog*, но в начало списка, чтобы наглядно показать, что эти задачи обязательны к выполнению в следующем цикле.

После внедрения Kanban произошло упрощение шага «*Ознакомление*», так как он стал частью этапа «*Планирование*». На рисунке 2 представлена новая модель процесса разработки ИС, где жирно выделены новые элементы.

Kanban не ограничивает участников команды. Абсолютно любую область разработки возможно визуализировать и разделить на этапы. Наряду с этим Kanban довольно простой инструмент с точки зрения внедрения.

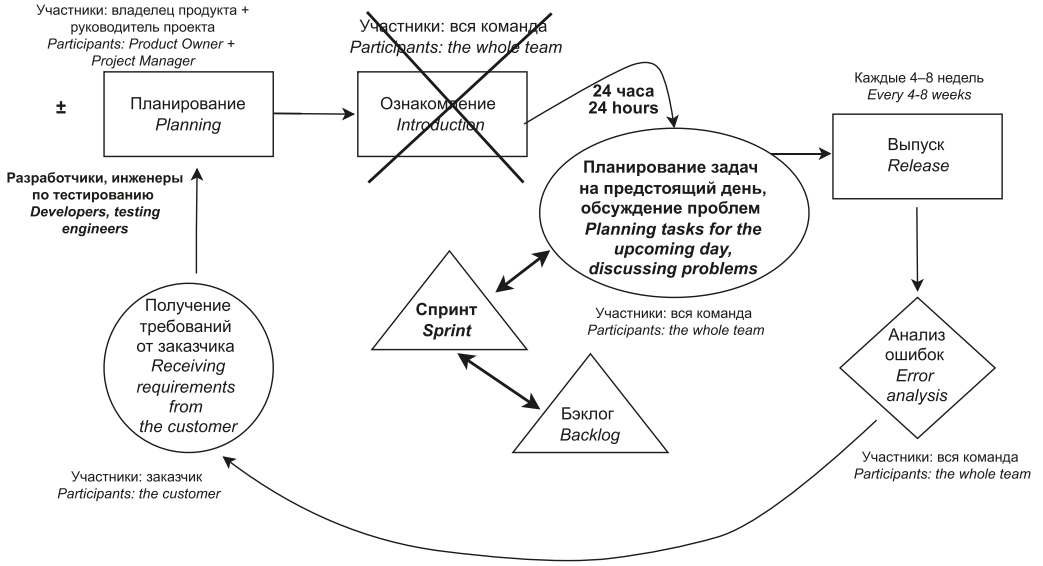

Рис. 2. Модель управления разработкой унаследованной системы на основе методики Kanban

Fig. 2. A management model for the development of a legacy system based on the Kanban methodology

Формирование UML-диаграммы прецедентов в качестве инструмента управления разработкой ПО

Разработка ПО с помощью методологии Kanban основана на выполнении большого списка задач, который формирован владельцем продукта. Описаны эти истории были в виде обычного линейного списка. Поэтому каждый раз во время планирования следующего цикла возникала необходимость просматривать весь этот список и формировать из него предстоящий объем работы.

Для того чтобы решить эту проблему, было предложено построить диаграмму вариантов использования (*Use Case Diagram*), которая помогла бы визуализировать поведение системы и распределить весь список пользовательских историй по соответствующим узлам.

Основными элементами диаграммы являлись:

1. *Участник (Actor).* Главное действующее лицо. Участником на диаграмме были обозначены как конечные пользователи, так и другие системы, взаимодействующие с прецедентом.
2. *Прецедент (Use Case).* Описание последовательности событий, выполняемых системой. Прецедент отображает поведение системы путем описания взаимодействий между системой и участниками. Этот элемент обозначает, что именно выполняется на данном этапе работы системы.
3. *Интерфейс (Interface).* Описание внешних параметров модели, которые доступны без описания внутренней структуры и реализации. Эти элементы являются стыковочными узлами в модели.
4. *Примечание (Note).* Элемент, требуемый для включения в модель необходимой текстовой информации, относящейся к контексту разрабатываемой системы.

Вначале были определены основные функциональные блоки разрабатываемой системы. После этого для каждого блока была построена диаграмма использования, на которой были отражены соответствующие функции в виде прецедентов. Для отображения параметров каждой функции использовались интерфейсы. Участником на диаграмме использования были смежные системы, так как конечный пользователь не взаимодействовал напрямую с разрабатываемой системой.

Разработка данной модели осуществлялась на этапе проектирования системы после того, как был сформирован список необходимых требований. Диаграмма использования впоследствии была доступна всем участникам процесса, включая самого заказчика. Используя построенную модель, заказчик совместно с владельцем продукта формировали требования на предстоящий цикл разработки.

Построенная диаграмма прецендентов была основой для формирования списка задач. Определенные изначально функциональные блоки представляли собой сущность *Solution Epic* с точки зрения имплементации проекта и были наивысшим уровнем его детализации.

Сущность *Solution Epic* представляет собой масштабные инициативы, требующие анализа, но связанные одним общим решением. Иными словами, *Solution Epic* – это масштабный проект, для реализации которого необходимо несколько команд и множество итераций или циклов. Каждая диаграмма прецедентов описывала основные этапы процесса разработки (*Epic*), распределенные сверху вниз на основе порядка их реализации.

Таким образом, распределив линейный список задач по узлам модели прецедентов, владелец продукта получал удобную и логичную структуру, работать с которой было намного проще. Полная модель прецедентов с распределенными задачами представлена на рисунке 3.

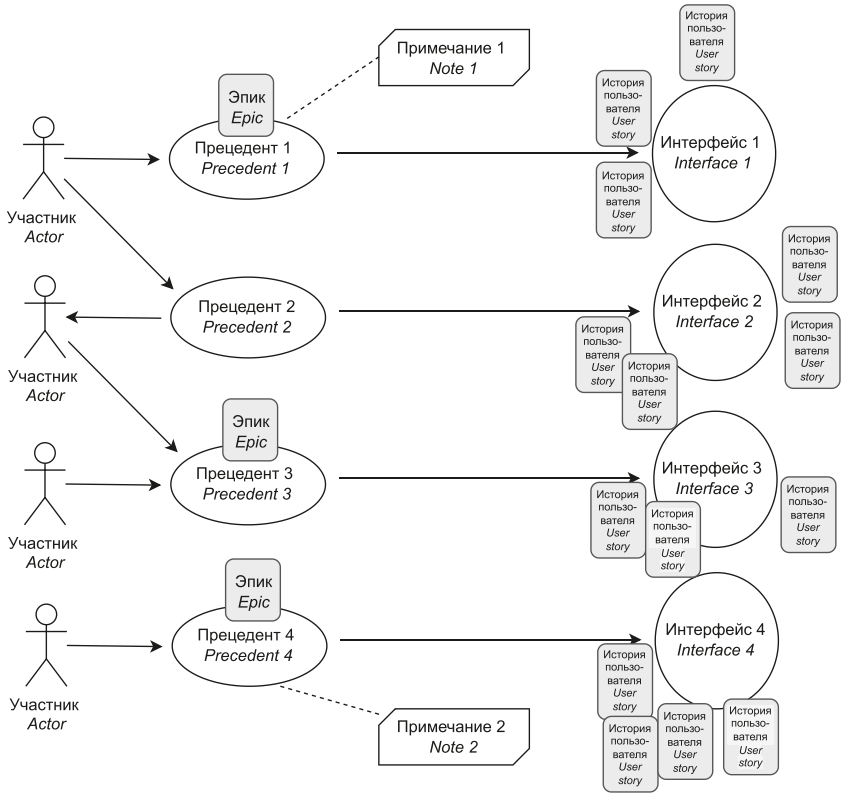

Рис. 3. Модель формирования списка задач на основе UML-диаграммы прецедента

Fig. 3. A model for generating a task list based on a UML precedent diagram

Основным преимуществом данной модели являлось то, что хаотичный список задач был разделен и структурирован в группы (*Epic*). Кроме того, это решение помогало визуализировать этапы реализации новых функций системы. Выделив в работе системы основные верхнеуровневые функции, а затем декомпозировав их на более мелкие единицы, владелец продукта получает полное представление о ходе процесса разработки.

До составления данной модели в проекте разработки ПО отсутствовала структура требований, именно поэтому никто из участников процесса не подозревал о необходимости доработки функциональности.

Созданная универсальная модель управления процессом разработки ПО телекоммуникационной системы основана на применении языка UML. Основной задачей при построении модели было графическое представление проекта, понятного участникам команды, отвечающим как за техническую, так и за бизнес-составляющую процесса разработки ПО.

Для того чтобы упростить и стандартизировать построение моделей архитектуры будущего приложения, был создан унифицированный язык программирования – UML (Unified Modelling Language). С помощью UML-языка возможно не только описывать и визуализировать архитектуру приложения, но и конструировать и документировать иные артефакты ИС.

Язык UML обладает рядом преимуществ, обуславливающих его популярность. Во-первых, UML дает возможность познакомиться с задачей с разных ее сторон и изучить детали реализации. Во-вторых,

синтаксис UML довольно простой, что позволяет достаточно быстро освоить его для полноценного использования в работе. В-третьих, диаграммы, построенные на языке UML, являются простыми для чтения. Поверхностного ознакомления с синтаксисом языка достаточно для освоения навыка чтения UML-диаграмм. В-четвертых, в UML существует порядка 15 видов диаграмм, что позволяет применять его в построении диаграмм для систем различной сложности и бизнес-областей.

Учитывая его гибкость, а именно совмещение с основной частью языков разработки, язык UML находит свое применение не только на этапе проектирования ПО. Такое решение можно назвать универсальным, так как оно подходит как для проектов, использующих традиционные методы разработки, так и для проектов, построенных с применением гибких методологий.

Сопоставив результат использования UML-диаграммы в качестве инструмента управления разработкой, можно сказать, что диаграммы UML позволяют не только визуализировать архитектуру системы на этапе ее построения, но и продумать требования и обсудить недостатки с коллегами и заказчиком. Диаграмма прецедентов довольно проста для понимания и подробно отражает состояние процесса разработки системы. Для заказчика такая модель является хорошим решением для понимания сути процесса разработки ПО.

По мере увеличения функциональности системы вносились и изменения в диаграмму прецедентов. Немаловажной задачей в данном случае являлась поддержка актуальности диаграммы. После каждого этапа сбора требований владелец продукта должен был доносить информацию до архитекторов, а те, в свою очередь, должны были документировать все изменения в системе и вносить изменения в диаграмму.

Главным преимуществом использования UML оказалась визуализация процесса разработки. Такой подход помог избавиться от хаотичного линейного списка задач и структурировать все требования, основываясь на бизнес-процессах, которые должна реализовывать система.

Стоит отметить, что использование диаграммы прецедентов в качестве инструмента управления разработкой помогло обнаружить недостатки в архитектуре системы, а также построить полную картину хода разработки.

Учитывая тот факт, что язык UML является доступным для понимания, построенная диаграмма была понятной всем участникам команды, включая заказчика, что оказало позитивное влияние на ход разработки ПО и дало заказчику четкое представление о состоянии проекта.

Заключение

Разработка программного обеспечения является достаточно сложным процессом, требующим большого количества ресурсов, не только денежных, но и временных, и трудовых. Повышенный спрос на разработку программного обеспечения, продиктованный цифровой трансформацией экономики, явился той отправной точкой, которая запустила процесс популяризации гибких методологий управления проектами во всех сферах экономической деятельности. Как справедливо было отмечено ранее, имеющиеся в настоящее время гибкие методологии управления проектами не являются унифицированными, что оказывает существенное влияние на их практическое применение в различных сферах экономики. Проведенное исследование позволило сформулировать основные критерии унификации гибких методологий и провести тестирование унифицированной модели на примере разработки программного обеспечения в телекоммуникационной компании.

В работе была описана и учтена специфика этапов внедрения гибких методоло-

гий в практику управления разработкой ПО в рамках эксперимента над одной из команд разработчиков телекоммуникационной компании ООО «ДОЙЧЕ ТЕЛЕКОМ АЙТИ РУС».

После внесения изменений в процесс управления разработкой ПО произошла трансформация управленческой модели, основанной на применении Kanban-методики (см. рис. 2). В частности, произошло упрощение фаз разработки, что повлекло за собой высвобождение временных, трудовых и денежных ресурсов. Полученный усовершенствованный процесс управления разработкой ПО может выступать основой для модернизации и унификации всей системы. Полученные результаты позволили определить, что тестируемая управленческая модель может быть применена не только в телекоммуникационной, но и в других экономических сферах.

Список литературы

1. *Абдалов М.С., Трубин А.Е., Нечаев А.М., Чаусов Д.Н.* Оценка эффективности применения технологий моб-программирования в управлении IT-проектами // Современная конкуренция. 2023. Т. 17. № 4. С. 80–89. DOI: 10.37791/2687-0649-2023-17-4-80-89.
2. *Анисимов А.Ю., Токмакова Е.Н., Трубин А.Е.* Компаративный анализ гибких методологий разработки программного обеспечения в условиях современных высококонкурентных рынков // Современная конкуренция. 2024. Т. 18. № 5. С. 111–123. DOI: 10.37791/2687-0657-2024-18-5-111-123.
3. *Богданов А.Е., Давлеткиреева Л.З.* Применимость методологии Kanban для управления небольшими IT-проектами с мелкими командами разработчиков // Глобальная экономика в XXI веке: роль биотехнологий и цифровых технологий: сборник научных статей. – М., 2020. С. 24–28.
4. *Васильева А.Д., Буторин А., Котегова Л.* Технология управления проектами на основе методологии Agile // Вестник Алтайской академии экономики и права. 2021. № 1-2. С. 118–124.
5. *Гришаева И.Н., Вайтекунайте П.Ю.* Использование гибких моделей управления командами ИТ-проектов // Управление человеческими ресурсами – основа развития инновационной экономики: материалы X Международной научно-практической конференции. – Красноярск, 2021. С. 250–254. DOI: 10.53374/9785864338810_25.
6. *Козлов С.В., Иванова М.В.* Основные принципы применения технологии экстремального программирования при разработке программного обеспечения // Современное состояние и перспективы развития науки и образования: сборник научных трудов. – Анапа, 2021. С. 196–205.
7. *Коломыцева А.О., Белоусов В.* Современные технологии управления проектами информатизации на основе методологий Agile-Scrum и Waterfall // Новое в экономической кибернетике. 2017. № 4. С. 106–118.
8. *Метельская Ю.Н., Шафранович П.С., Кашникова И.В.* Использование Scrum, Kanban в проектах гибкой разработки программного обеспечения для различных сфер деятельности организаций // Роль и место инноваций в сфере агропромышленного комплекса: материалы Всероссийской (национальной) научно-практической конференции. – Курск, 2020. С. 419–423.
9. *Онокой Л.С.* Гибкие подходы к разработке программного обеспечения: эволюция и перспективы развития // Качество. Инновации. Образование. 2021. № 1 (171). С. 57–67. DOI: 10.31145/1999-513x-2021-1-56-66.
10. *Раубецкий А.В.* Технология управления проектами и проектными командами на основе методологии гибкого управления проектами Agile // Научные исследования в современном мире: опыт, проблемы и перспективы развития: сборник научных статей. – Уфа, 2023. С. 57–66.
11. *Родина М.А.* Экстремальное программирование (XP): подходы и опыт «гибкого» внедрения в современных компаниях // Взгляд молодых ученых на проблемы устойчивого развития: сборник научных статей. 2019. Т. 8. С. 142–148.
12. *Торосян Е.К., Тюлькина А.С.* Критерии выбора методологии управления IT-проектами // Петербургский экономический журнал. 2020. № 1. С. 99–108. DOI: 10.25631/PEJ.2020.1.99.10.

13. *Ямщикова А. Е., Никульников Н. В.* Гибкая разработка, или инновационные технологии управления проектами на основе Agile // Вопросы экономических наук. 2022. № 1 (113). С. 54–57.
14. *Cohen E.* The Definitive Guide to Project Management Methologies // Workamajig. URL: https://www.workamajig.com/blog/project-management-methodologies (дата обращения: 29.10.2024).

Сведения об авторах

Анисимов Александр Юрьевич, ORCID 0000-0002-8113-4523, канд. экон. наук, доцент, заместитель директора по учебно-методической работе факультета информационных технологий, доцент кафедры информационного менеджмента им. профессора В. В. Дика, Университет «Синергия», Москва, Россия, anisimov_au@mail.ru

Трубин Александр Евгеньевич, ORCID 0000-0002-7189-5679, канд. экон. наук, доцент, заведующий кафедрой цифровой экономики, Университет «Синергия», Москва, Россия, niburt@yandex.ru

Токмакова Елена Николаевна, ORCID 0000-0001-9963-2726, канд. экон. наук, доцент, доцент кафедры цифровой экономики, Университет «Синергия», Москва, Россия, e_tokmakova@mail.ru

Филимонова Елена Викторовна, ORCID 0000-0002-9791-7610, канд. пед. наук, доцент, кафедра цифровой экономики, Университет «Синергия», Москва, Россия, elena-gamilton@mail.ru

Статья поступила 05.11.2024, рассмотрена 19.11.2024, принята 05.12.2024

References

1. Abdalov M., Trubin A., Nechaev A., Chausov D. Performance Evaluation of Application Mob Programming Technologies in IT Project Management. *Sovremennaya konkurentsiya*=Journal of Modern Competition, 2023, vol.17, no.4, pp.80-89 (in Russian). DOI: 10.37791/2687-0649-2023-17-4-80-89.
2. Anisimov A., Tokmakova E., Trubin A. Comparative Analysis of Flexible Software Development Methodologies in Modern Highly Competitive Markets. *Sovremennaya konkurentsiya*=Journal of Modern Competition, 2024, vol.18, no.5, pp.111-123 (in Russian). DOI: 10.37791/2687-0657-2024-18-5-111-123.
3. Bogdanov A.E., Davletkireeva L.Z. *Primenimost' metodologii Kanban dlya upravleniya nebol'shimi IT-proektami s melkimi komandami razrabotchikov* [Applicability of the Kanban methodology for managing small IT projects with small development teams]. *Global'naya ekonomika v XXI veke: rol' biotekhnologii i tsifrovykh tekhnologii: sbornik nauchnykh statei* [Global Economy in the 21st Century: The Role of Biotechnologies and Digital Technologies: Collection of Scientific Articles]. Moscow, 2020, pp.24-28.
4. Vasilyeva A.D., Butorin A., Kotegova L. Project management technology based on the Agile methodology. *Vestnik Altaiskoi akademii ekonomiki i prava*, 2021, no.1-2, pp.118-124 (in Russian).
5. Grishaeva I.N., Vaitekunaite P.Yu. Using flexible models of IT-project team management. *Upravlenie chelovecheskimi resursami – osnova razvitiya innovatsionnoi ekonomiki: materialy X Mezhdunarodnoi nauchno-prakticheskoi konferentsii* [Human Resource Management – The Basis for the Development of an Innovative Economy: Materials of the X International Scientific and Practical Conference]. Krasnoyarsk, 2021, pp.250-254 (in Russian). DOI: 10.53374/9785864338810_25.
6. Kozlov S.V., Ivanova M.V. *Osnovnye printsipy primeneniya tekhnologii ekstremal'nogo programmirovaniya pri razrabotke programmnogo obespecheniya* [Basic principles of using Extreme Programming technology in software development]. *Sovremennoe sostoyanie i perspektivy razvitiya nauki i obrazovaniya: sbornik nauchnykh trudov* [Current state and prospects for the development of science and education, collection of scientific papers]. Anapa, 2021, pp.196-205.
7. Kolomysheva A.O., Belousov V.V. Modern technologies and management of informatization projects based on methodologies Agile-Scrum and Waterfall. *Novoe v ekonomicheskoi kibernetike*, 2017, no.4, pp.106-118 (in Russian).

8. Metelskaya Yu.N., Shafranovich P.S., Kashnikova I.V. *Ispol'zovanie Scrum, Kanban v proektakh gibkoi razrabotki programmnogo obespecheniya dlya razlichnykh sfer deyatel'nosti organizatsii* [The use of Scrum, Kanban in flexible software development projects for various areas of activity of organizations]. *Rol' i mesto innovatsii v sfere agropromyshlennogo kompleksa: materialy Vserossiiskoi (natsional'noi) nauchno-prakticheskoi konferentsii* [The role and Place of Innovation in the Field of Agro-industrial Complex: Materials of the All-Russian (National) Scientific and Practical Conference]. Kursk, 2020, pp.419-423.
9. Onokoy L.S. Flexible approaches to the development of software: Evolution and prospects of development. *Kachestvo. Innovatsii. Obrazovanie*=Quality. Innovation. Education, 2021, no.1(171), pp.56-66 (in Russian). DOI: 10.31145/1999-513x-2021-1-56-66.
10. Raubetsky A.V. Project and project team management technology based on agile project management methodology. *Nauchnye issledovaniya v sovremennom mire: opyt, problemy i perspektivy razvitiya: sbornik nauchnykh statei* [Scientific Research in the Modern World: Experience, Problems and Development Prospects: Collection of Scientific Articles]. Ufa, 2023, pp.57-66 (in Russian).
11. Rodina M.A., Mukhin K.Yu. Extreme Programming (XP): Approaches and best practices of "Agile" implementing to modern companies. View of Young Scientists on the Problems of Sustainable Development: Collection of Scientific Articles. 2019, vol.8, pp.142-148 (in Russian).
12. Torosyan E.K., Tyulkina A.S. Selection criteria for IT project management. *Peterburgskii ekonomicheskii zhurnal*=St. Petersburg Economic Journal, 2020, no.1, pp.99-108 (in Russian). DOI: 10.25631/PEJ.2020.1.99.10.
13. Yamshchikova A.E., Nikulnikov N.V. *Gibkaya razrabotka, ili innovatsionnye tekhnologii upravleniya proektami na osnove Agile* [Flexible development, or innovative project management technologies based on Agile]. *Voprosy ekonomicheskikh nauk*, 2022, no.1(113), pp.54-57.
14. Cohen E. The Definitive Guide to Project Management Methologies. Workamajig. Available at: https://www.workamajig.com/blog/project-management-methodologies (accessed 29.10.2024).

About the authors

Alexander Yu. Anisimov, ORCID 0000-0002-8113-4523, Cand. Sci. (Econ.), Assistant Professor, Deputy Director for Educational and Methodological Work of Information Technologies Faculty, Information Management Department named after Professor V. V. Dick, Synergy University, Moscow, Russia, anisimov_au@mail.ru

Alexander E. Trubin, ORCID 0000-0002-7189-5679, Cand. Sci. (Econ.), Associate Professor, Director of Digital Economy Department, Synergy University, Moscow, Russia, niburt@yandex.ru

Elena N. Tokmakova, ORCID 0000-0001-9963-2726, Cand. Sci. (Econ.), Associate Professor, Digital Economy Department, Synergy University, Moscow, Russia, e_tokmakova@mail.ru

Elena V. Filimonova, ORCID 0000-0002-9791-7610, Cand. Sci. (Ped.), Associate Professor, Digital Economy Department, Synergy University, Moscow, Russia, elena-gamilton@mail.ru

Received 05.11.2024, reviewed 19.11.2024, accepted 05.12.2024