УДК 004.9:412

Ю. В. ЖМАЕВА, С.Г. УДОВЕНКО, Л.Э. ЧАЛАЯ

АДАПТИВНОЕ ПРОГНОЗИРОВАНИЕ ОПТИМАЛЬНОГО КОЛИЧЕСТВА РЕСУРСОВ IT ПРОЕКТА ПО МЕТОДОЛОГИИ AGILE

Предлагается метод расчета и прогнозирования оптимального количества ресурсов IT проекта по методологии Agile. Данный метод разрабатывается для управления риск менеджментом при проектировании программного обеспечения. Осуществляется программная реализация метода на языке программирования Java с использованием Spring MVC Framework.

1. Введение и постановка задачи

Классические индикаторы эффективности разработки программного обеспечения связаны с утилизацией ресурсов, позволяющей сравнивать фактически выполненный и запланированный объем работ. Следует отметить, что такие индикаторы сложны в восприятии и применяются только в классических инструментах планирования (например, в методиках, основанных на использовании диаграмм Ганта).

В последнее время развиваются методы создания Agile проектов с применением более простых и наглядных индикаторов состояния процесса разработки (например, коэффициента возврата инвестиций, планируемых и затраченных ресурсов и т.п.). Однако локальное использование каждого из таких показателей не является достаточно информативным для динамического отображения степени эффективности проектирования. В то же время комплексное использование совокупности характеристик и метрик Agile позволяет прогнозировать скорость и качество выполнения проектных задач. Agile методологии, как правило, основаны на построении диаграммы выполнения задач, которая позволяет проанализировать существующие проблемы в момент разработки и устранить их для успешного завершения итерации. График прогресса разработки – широко используемый способ, с помощью которого команда отслеживает темп выполнения итерации. Концепция диаграммы выполнения задач состоит в том, чтобы в выбранной контрольной точке отобразить, как много задач осталось выполнить. Применение данного артефакта в сфере управления проектами дает возможность проанализировать узкие места на этапах планирования, разработки и поддержки приложения.

Существующий рынок информационных технологий предлагает различные системы отслеживания ошибок, оперирующие определенным набором характеристик для построения диаграмм производительности, совокупного потока, графиков прогресса разработки и др. [1, 2]. Однако даже самые современные системы предоставляют лишь часть необходимых инструментов для построения диаграммы выполнения задач (burn-down chart). Используемые здесь показатели (скорость команды, длительность итерации и сложность задач) не являются достаточными для отображения реальной динамики проектирования из-за неравномерности распределения работ между участниками проекта, выполняющими различные роли.

Решением этой проблемы может быть усовершенствование диаграммы с помощью расширения входных параметров и подходов для увеличения точности отображаемых результатов. Целесообразно разработать адаптивный метод ведения диаграммы, учитывающий показатели скорости команды на каждой фазе разработки. Индикатор такого метода должен отображать выполненный объем задач в каждой контрольной точке, а также разницу между текущей и требуемой скоростью команды.

Целью данного исследования является разработка и программная реализация метода расчета и адаптивного прогнозирования оптимального количества ресурсов IT проекта по методологии Agile. При разработке метода должны быть учтены как стандартные показатели, характеризующие человеческие ресурсы (рабочие часы, дни, длительность итерации, сверхурочные часы, степень участия того или иного члена команды в проекте, максимальная скорость команды), так и дополнительные показатели, позволяющие повысить наглядность

предсталения результатов (фокус-фактор, количество незапланированного объема работы, количество оставшейся работы по неделям, разница оценки задач). Целевой аудиторией предлагаемого подхода являются участники Agile команд, скрам-мастер, менеджер проекта, менеджер департамента, заказчик.

2. Общие принципы построения диаграммы выполнения задач

Гибкие методологии создания программного обеспечения ориентированы на использование итеративной разработки, динамического формирования требований и обеспечения их реализации в результате постоянного взаимодействия внутри самоорганизующихся рабочих групп, состоящих из специалистов различного профиля. Существует несколько методик, относящихся к классу таких гибких методологий, в частности экстремальное программирование, DSDM (Dynamic Systems Development Method), Scrum, FDD (Feature driven development). Эти методики основаны, как правило, на минимизации рисков путем сведения разработки к серии коротких циклов, называемых итерациями (длительность таких циклов часто принимается двухнедельной). Каждая итерация сама по себе представляет программный проект в миниатюре и включает все задачи, необходимые для выдачи мини-прироста по функциональности: планирование, анализ требований, проектирование, программирование, тестирование и документирование. Хотя отдельная итерация, как правило, недостаточна для выпуска новой версии продукта, подразумевается, что гибкий программный проект готов к выпуску в конце каждой итерации. По окончании каждой итерации команда выполняет переоценку приоритетов разработки. Диаграмма выполнения задач является самым распространенным и эффективным инструментом методологий Agile, отображающим динамику проектирования и используемым для контроля прогресса на любой итерации спринта [1]. На рис. 1 приведена диаграмма, отражающая классический график процесса разработки по методологии Agile.

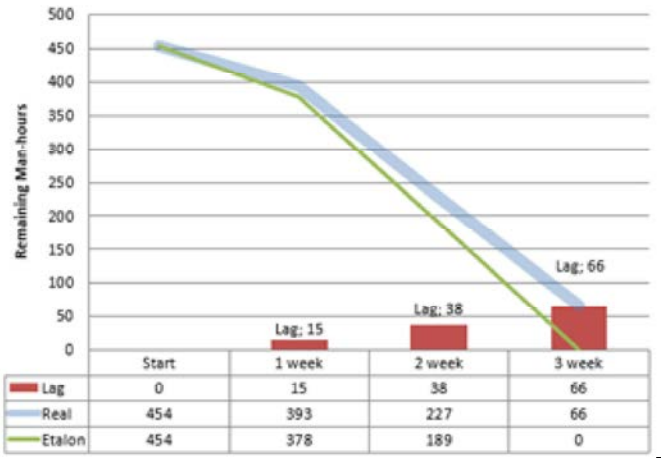

Рис. 1. Классический график процесса разработки

Основные составляющие классического графика процесса разработки: ось X отображает календарные дни итерации; ось Y – объем работы (может измеряться в пользовательских историях, человеко-часах, днях, количестве задач); зеленая кривая отображает ожидаемый прогресс; синяя кривая отображает реальный прогресс и количество оставшейся работы. Зеленая кривая выходит из точки, соответствующей началу итерации и исходному объему работ, а заканчивается на дате завершения итерации. Такой график выполнения задач командой является идеальным и показывает, сколько часов должно быть потрачено ежедневно для успешного выполнения всех задач к окончанию итерации. Реальный график выполнения задач отображается синей кривой, показывающей оставшуюся трудоемкость на каждый день итерации. Если кривая расположена под зеленой линией, то это означает, что команда движется с достаточной скоростью и успевает завершить задачи по итерации к намеченному сроку. Если синяя кривая располагается над зеленой линией, нужно либо повлиять на скорость команды, либо исключить из итерации какие-то задачи, поскольку иначе команда не успеет завершить работу в срок [3]. С помощью графика процесса разработки можно определить не только скорость команды, но и узкие места, препятству-

ющие достижению цели в поставленные сроки. Современные системы отслеживания ошибок поддерживают автоматическую генерацию диаграммы прогресса разработки. Jira – коммерческая система отслеживания ошибок, предназначенная для организации взаимодействия с пользователями. Основной элемент учета в этой системе – задача (ticket). Она содержит название проекта, тему, тип, приоритет, компоненты и содержание. Набор задач формирует базу данных для построения графика. Система Rational Team Concert (RTC) предоставляет график прогресса разработки и управляет жизненным циклом ПО, обеспечивая контекстную коллективную работу для распределенных команд в реальном времени. Система RTC упрощает планирование и выполнение гибких или формальных проектов на основе инструментов планирования и шаблонов. Согласованные процессы, основанные на применении этих систем, помогают повысить качество разрабатываемого программного обеспечения, однако не учитывают в должной мере все существенные различные факторы влияния на график. Кроме того, диаграмма показывает общую скорость команды, однако часто команды состоят из аналитиков, разработчиков, менеджеров и тестировщиков, неравномерно участвующих в процессе. В этом случае непонятно, в какой части процесса возникают трудности. Предлагаемый ниже метод позволяет осуществить усовершенствование диаграммы с помощью расширения входных параметров для увеличения точности отображаемых результатов.

3. Разработка метода построения диаграммы выполнения задач

Диаграмма выполнения задач должна содержать набор параметров, которые определяют тренд для построения графика. Такая диаграмма состоит из множества характеристик, используемых в методологиях Agile. Эталонный тренд, представленный на рис. 2, учитывает скорость команды на каждой календарной неделе и распределяет ориентир соответственно [4]. Кривая эталона показывает объем работы, который команда должна выполнить в каждой контрольной точке, чтобы успеть завершить все задачи вовремя.

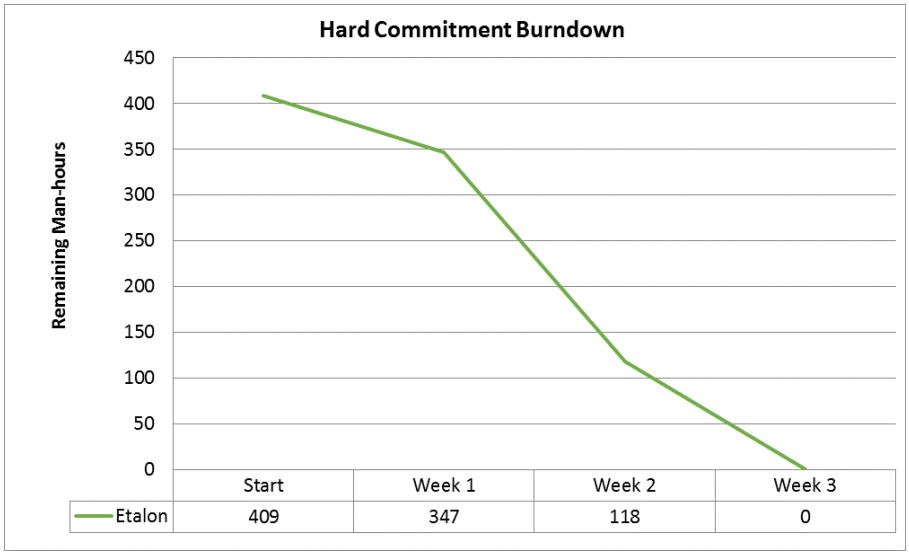

Рис. 2. Эталонный ориентир

В первую и последнюю календарную недели спринта ожидаемый объем выполненных работ намного ниже, чем во вторую, так как начинается и заканчивается спринт в середине недели. Показатели эталонного ориентира оцениваются в человеко-часах – единица учета рабочего времени, которая соответствует часу работы одного человека [5]. Данная единица позволяет оценить работу при планировании более точно, сопоставляя количество работников и сроки выполнения задания. Эталонный ориентир определяется по формуле:

$$E = \{E_{start}, E_{week1}, \dots, E_{weekN}\}, \quad (1)$$

где E_{start} – предполагаемый объем работы в начале итерации; E_{weekN} – объем работы в каждой контрольной точке;

$$E_{start} = S + AVG\Delta_{start}, \quad (2)$$

здесь S – объем работы в итерации; $AVG\Delta_{start}$ – объем незапланированных задач в начале итерации;

$$E_{weekN} = \frac{E_{weekN-1} - E_{start} \times C_{weekN}}{C_{sprint}}, \quad (3)$$

C_{weekN} – скорость команды в течение недели; C_{sprint} – скорость команды в итерации.

Каждая задача, планируемая к выполнению за определенный промежуток времени (итерацию), оценивается в идеальных часах. Данный параметр позволяет учитывать все процессы для точного расчета ресурсоемкости проекта на этапе планирования итерации. Время по задачам суммируется, образуя метрику Team Capacity (емкость, объем работы):

$$C_{period} = 8 \times WD * EN - V - SN + O, \quad (4)$$

где WD – количество рабочих дней; EN – количество участников проекта; V – количество дней отпуска; SN – количество дней, отведенных на больничный; O – количество часов сверхурочной работы.

При вычислении объема работ учитываются такие показатели, как количество рабочих дней, количество инженеров, отпуска, больничные, сверхурочные часы [6]. Таким образом, команде предоставляется точный индикатор того, с какой скоростью должны быть выполнены задачи.

Второй тренд Scope отражает фактическое выполнение задач. Данный показатель, представленный на рис. 3, показывает как команда справляется с задачами.

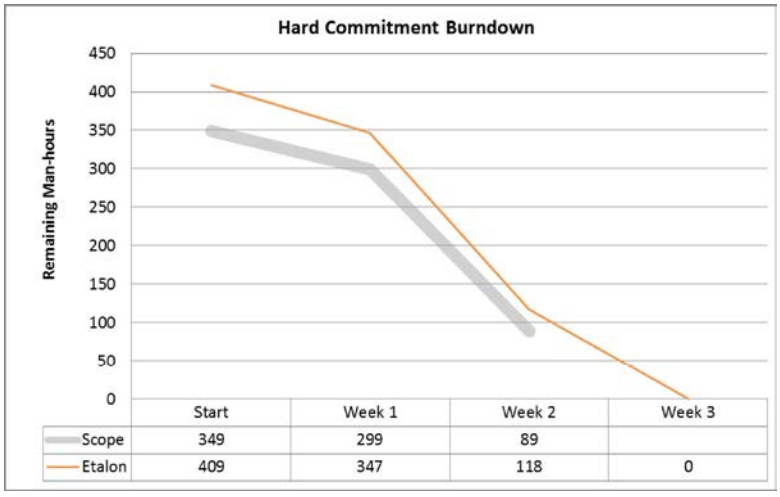

Рис. 3. Выполнение задач на протяжении итерации

В каждой задаче указано, сколько времени уже потрачено на ее реализацию (spent) и сколько еще осталось потратить (remaining). Сумма всех remaining-показателей образует серую линию и определяется по формуле:

$$S = \sum RV, \quad (5)$$

где RV – количество оставшихся человеко-часов.

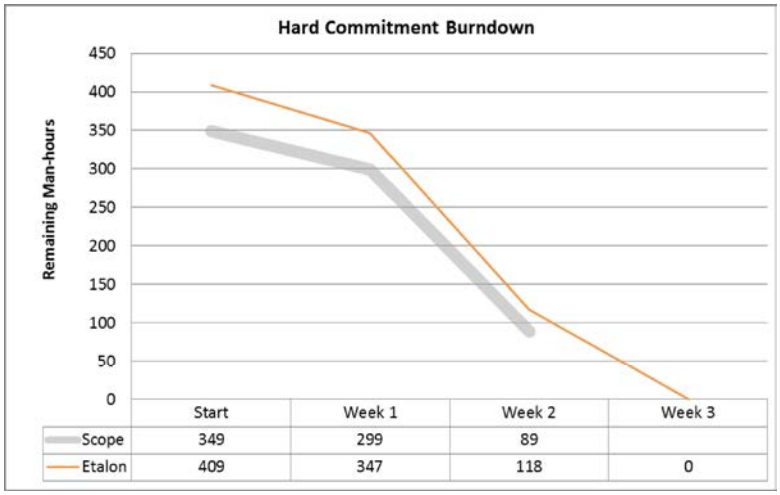

Рис. 3. Выполнение задач на протяжении итерации

В классическом виде такая модель недостаточно показательна. Если команда будет руководствоваться только текущим состоянием командной доски, производительность команды не будет отображена в полной мере, так как командная доска, содержащая задачи для текущей итерации, не отражает ряд незапланированных активностей, которые должны быть выполнены (например, исправление дефектов по комментариям от инженера по контролю качества кода и сопутствующее регрессионное тестирование) [7]. Исполнитель имеет также право отклониться от оценок, определенных им ранее. При этом объем запланированных работ к концу итерации всегда будет больше изначального набора задач.

Если команда не будет учитывать прогноз объема незапланированных задач, представленных на рис. 4, то это может привести к возникновению рисков потери качества продукта.

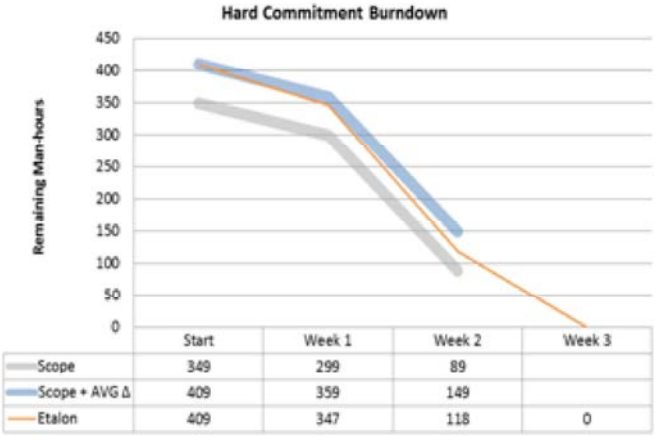

Рис. 4. Нормализированный объем незапланированных задач

Синяя кривая – это показатель нормализированного объема оставшейся работы, в состав которого включен прогноз на незапланированные задачи и превышение оценок. Этот показатель определяется следующим образом:

$$S + AVG\Delta = S * UTC * EDC, \quad (6)$$

где UTC – количество незапланированных задач; EDC – мера превышения оценок.

Прогноз незапланированных задач и превышения оценок осуществляется для расчета наиболее точного ожидаемого результата в конце итерации. Современные системы отслеживания ошибок не учитывают данные показатели, которые значительно могут повлиять на работу команды. Если прогноз на незапланированный объем задач не будет учтен, это может способствовать возникновению рисков, которые повлекут за собой: снижение качества раз-

рабатываемого продукта; увеличение ресурсов, рабочих часов, что в свою очередь повлияет на финансовую составляющую проекта; расторжение контракта с заказчиком; несоблюдение Agile методологии.

Прогнозируемые характеристики определяются как взвешенное среднее результатов предыдущих спринтов. Вес конкретной итерации определяется консолидированной оценкой, руководствующейся набором адаптивных правил. В показатель включается вес всех незапланированных задач. Прогноз незапланированного объема задач определяется по формуле:

$$US_{forecast} = \frac{S \times (US_{AVG} - US_{current})}{100\%}, \quad (7)$$

где US_{AVG} – взвешенное среднее результатов предыдущих спринтов; $US_{current}$ – количество незапланированных задач в текущей итерации.

Объем незапланированных задач для текущей итерации определяется следующим образом:

$$US_i = \frac{\sum RV_i}{S + AVG\Delta_i \times 100\%}, \quad (8)$$

здесь $\sum RV_i$ – количество незапланированных задач в текущей итерации.

Взвешенное среднее результатов предыдущих спринтов определяется по формуле:

$$US_{AVG} = \frac{US_i \times USW_i}{\sum USW_i}, \quad (9)$$

где USW_i – коэффициент незапланированных задач.

Вторым показателем, влияющим на прогноз, является отклонение от ошибки. Каждая задача на командной доске содержит изначальную оценку, которая была проставлена исполнителем для выполнения данной задачи, и фактически потраченное время.

Сумма разниц этих показателей отображает превышение изначальной суммарной оценки. Процедура корректировки коэффициента влияния определяется для конкретной итерации индивидуально по таким критериям: нахождение наиболее аномальных превышений оценки; выявление причин превышения; определение превентивных мер.

Прогноз превышения оценок находится по формуле:

$$ED_{forecast} = \frac{(S + US_{forecast}) \times ED_{AVG}}{100\%}, \quad (10)$$

где ED_{AVG} – взвешенное среднее результатов предыдущих спринтов.

Погрешность в оценке в текущей итерации вычисляется по формуле:

$$ED_i = \left(\frac{\sum SV + RV - OV}{S_i}\right) \times 100\%, \quad (11)$$

здесь SV – потраченное время на выполнение задачи; RV – оставшееся время; OV – запланированное время.

Отклонение от оценки в текущей итерации определяется по формуле:

$$ED_{AVG} = \frac{ED_i \times EDW_i}{\sum EDW_i}, \quad (12)$$

где EDW_i – коэффициент превышения оценок.

По результатам построения эталонной и действительной тенденций рассчитываются основные показатели эффективности команды [8]. Первый показатель демонстрирует разницу между запланированным и выполненным объемом, т.е. насколько команда отстает от графика или его опережает (с учетом прогноза на отклонение). Пример применения показателя эффективности команды представлен на рис. 5.

Данный параметр (в человеко-часах) определяется следующей суммой:

$$LAG = S + AVG\Delta - E. \quad (13)$$

Если команда выполняет задачи равномерно, следуя плану, то отклонение от графика будет равняться нулю. В случае, когда параметр Lag отрицательный, то это сигнализирует

о том, что команда опережает план. Если же он положительный, это может свидетельствовать о том, что команда не успевает выполнить работу в поставленный срок. Данную ситуацию можно объяснить следующими причинами: увеличение сложности одной или нескольких пользовательских историй; сокращение ресурсов; ошибка в изначальной оценке на этапе планирования итерации; нахождение большого количества дефектов после фазы тестирования; увеличение объема смежного функционала, который был задействован в момент разработки; недостаток опыта и знаний в команде для решения поставленных задач.

	Start	Week 1	Week 2	Week 3
Lag	0	12	31	
Scope	349	299	89	
Scope + AVG Δ	409	359	149	
Etalon	409	347	118	0

Рис. 5. Эффективность с учетом прогноза на отклонение

Данный показатель необходим команде для устранения рисков на этапе их формирования. Причины возникновения рисков могут быть выявлены с помощью таких показателей как фокус-фактор и количество человеко-часов в текущей итерации. Описанная выше модель позволяет команде контролировать выполнение задач в итерации, но не отображает эффективность расхода командой временных ресурсов. Фокус-фактор (focus factor) – это индикатор, показывающий, какую часть времени команда тратит на непосредственное выполнение запланированных задач. Данный показатель может быть смещен различными шумами (например, неспособностью или нежеланием команды фиксировать потраченное время на каждом этапе разработки). Фокус-фактор рассчитывается по формуле:

$$FF = \frac{WD}{C} \times 100\%, \tag{14}$$

где WD – количество человеко-часов, которые были потрачены на реализацию; C – скорость команды.

Нижняя допустимая граница командного фокус-фактора в Agile методологиях принимается равной 75%. Это означает, что команда должна тратить более s рабочего времени на выполнение задач на командной доске для успешной реализации пользовательской истории. Графическое отображение примера командного фокус-фактора представлено на рис. 6.

Чтобы наиболее точно идентифицировать причину низкого командного фокус-фактора, следует учитывать данный параметр для разработчиков, тестировщиков и других членов команды отдельно. Например, общий показатель может быть меньше 75% по причине недостаточной эффективности работы бизнес-аналитиков. В таком случае менеджер точечно определит проблему, вызывающую отставание, и устранит ее. Дополнительным индикатором, отображающим эффективность команды, является скорость выполнения пользовательских историй.

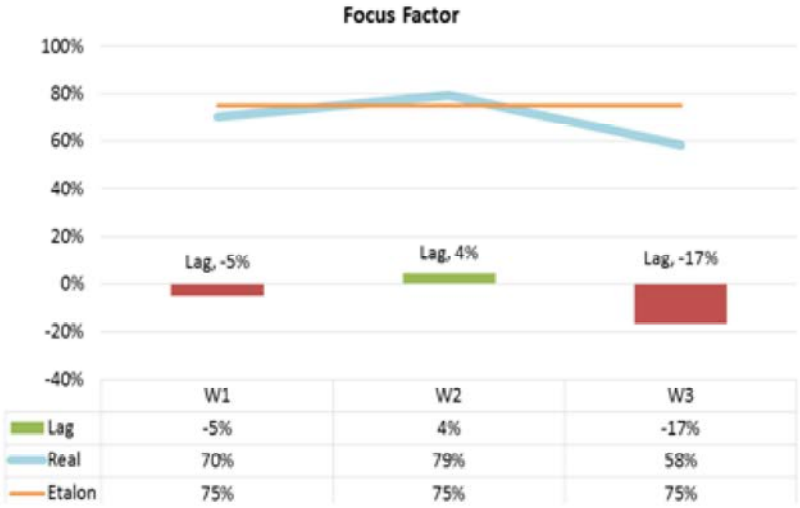

Рис. 6. Пример командного фокус-фактора

Для наглядности данный параметр рассматривается с учетом максимальной, эталонной и действительной скорости команды. Максимальным ресурсам, на которые может рассчитывать команда, соответствует полное привлечение сторонних разработчиков, тестировщиков и других участники проекта. Эталонный показатель учитывает лишь текущие ресурсы, без учета неполной занятости или отсутствия некоторых участников команды. Действительная скорость – скорость в текущей итерации, в которой были учтены больничные, отпуска, сверхурочные часы.

4. Программная реализация

Для реализации поставленной задачи было разработано Web-приложение на основе схемы использования шаблонов MVC, позволяющее разделять данные, представление и обработку действий пользователя на три отдельных компонента: модель, представление, вид. Для создания более гибкой архитектуры был использован Spring Framework, обеспечивающий решение многих задач для создания информационной системы с применением языка Java. При работе с приложением пользователю предоставляется возможность использовать различные формы для заполнения данных: количество участников команды, роль и занятость, количество рабочих дней, даты контрольных точек. Пример создания новой итерации представлен на рис.7.

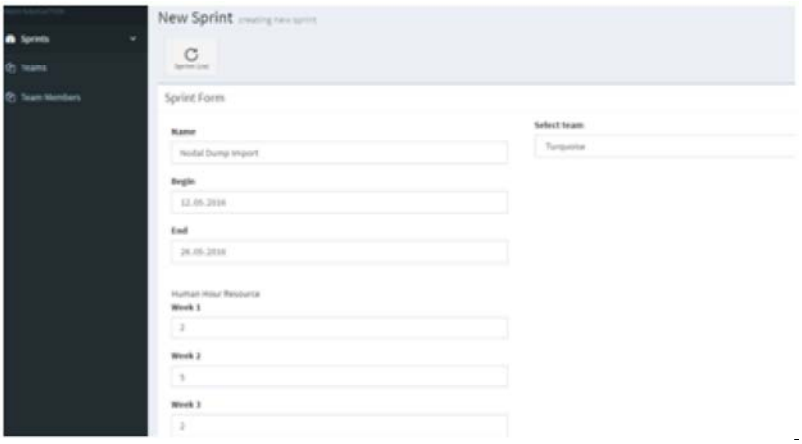

Рис. 7. Форма создания спринта

К созданной итерации пользователь может добавить список проектных задач, содержащий характеристики каждой записи. Например, страница «Sprint Info», представленная на рис. 8, содержит информацию о текущей итерации: количество участников и их роли в

итерации, расчет текущих ресурсов, скорость команды, объем запланированной и выполненной работы, фокус-фактор, прогноз на незапланированный объем задач и погрешность в оценке, график выполнения задач. Также на странице представлен прогноз объема незапланированных задач и превышения оценок. Результаты прогноза отображаются, начиная с третьей итерации, когда система хранит достаточное количество данных для прогнозирования. С каждой итерацией оценка показателей эффективности становится точнее. Данные показатели можно проанализировать в любой фазе итерации, что делает модель гибкой и легко масштабируемой.

Рис. 8. Объем выполненных задач на протяжении итерации

На странице «Graphics» представляется диаграмма выполнения задач, в построении которой участвуют все атрибуты и параметры, описанные выше (рис. 9).

Рис. 9. График выполнения задач

Ось Y отображает количество человеко-часов, ось X – контрольные точки, которые были определены при создании итерации согласно рабочим неделям. Первая контрольная точка – первый день итерации, четвертая – последний. Пользователю предоставлена возможность корректировать даты контроля для мониторинга эффективности на любой фазе спринта. Диаграмма выполнения задач состоит из трех кривых и показателя Lag. Серая кривая на графике отображает фактическое выполнение задач. Данный показатель отображает, с какой скоростью команда выполняет проектные задачи согласно доске. В каждой задаче указано, сколько времени уже потрачено на ее реализацию и сколько осталось потратить. Сумма всех человеко-часов, необходимых для выполнения задачи, соответствует серой кривой. Эталонный ориентир представляется зеленой кривой, которая показывает, какой объем работ команда должна завершить в каждой

контрольной точке, чтобы успеть выполнить все задачи вовремя. Эталонный тренд учитывает скорость команды на каждой календарной неделе и распределяет ориентир соответственно. Синяя кривая соответствует нормализированному объему оставшейся работы, в состав которого включен прогноз на незапланированные задачи и превышение оценок. Последний показатель Lag, представленный на рис. 9, наглядно демонстрирует разницу между эталонной и действительной тенденцией, т.е. насколько команда отстает или же опережает график с учетом прогноза на отклонение. Пользователю предоставляется возможность сохранить график в различных форматах: PNG, JPEG, PDF, SVG. Таким образом, документ может быть распечатан так, как он отражается на экране пользователя. Кроме того, генерация PDF файлов является удобным способом документирования. Разработанный программный продукт характеризуется открытостью, кроссплатформенностью и распространенностью, обеспечивая точность и неизменность передачи данных по цепочке «генерация - просмотр – печать».

5. Выводы и перспективы дальнейших исследований

Современные системы отслеживания ошибок основаны на различных способах визуализации хода разработки программного обеспечения для IT проектов. Управление проектами представляет собой последовательный, поступательный способ координации деятельности в сфере разработки программного обеспечения и информационных технологий. Основой построения плана команды в гибких методологиях разработки Agile является динамическое формирование диаграммы выполнения задач.

Предложенный метод позволяет усовершенствовать диаграммы с помощью расширения входных параметров для увеличения точности отображаемых результатов. Прогнозируемые характеристики определяются при этом как взвешенное среднее результатов предыдущих спринтов. Вес конкретной итерации устанавливается консолидированной оценкой, руководствующейся набором адаптивных правил. В показатель включается вес всех незапланированных задач.

Разработанное для реализации предложенного метода Web-приложение на основе схемы использования шаблонов MVC позволяет осуществлять представление и обработку действий пользователя по трем отдельным компонентам (модель, представление, вид). Для создания более гибкой архитектуры был использован Spring Framework, обеспечивающий решение многих задач для создания информационной системы с применением языка Java.

Перспективным развитием рассмотренного подхода является расширение его функциональности на основе комплексного применения наиболее перспективных методологий управления IT проектами, что позволит избавиться от привязки к конкретным видам разработки.

Список литературы: 1. *Cohn, M.* Succeeding with Agile: Software Development Using Scrum [Текст] / Cohn, M. W.: Pearson Education, 2009. 114 p. **2.** *Martin R. C.* Agile Software Development, Principles, Patterns, and Practices [Текст] / Martin, R. C. N.: Pearson Education, 2002. 458 p. **3.** *Rubin K. S.* Essential Scrum: A Practical Guide to the Most Popular Agile Process [Текст] / Rubin K. S. H.: Addison-Wesley Professional, 2012. 281 p. **4.** *Maurya, A.* Running Lean: Iterate from Plan A to a Plan That Works [Текст]/ Maurya, A. W.: O'Reilly Media, 2012. 129 p. **5.** *Ries E.* The Lean Startup: How Today's Entrepreneurs Use Continuous Innovation to Create Radically Successful Businesses [Текст] / Ries, E. W.: Crown Business, 2011. 90 p. **6.** *Croll A.* Lean Analytics: Use Data to Build a Better Startup Faster [Текст] / Croll, A. R.: O'Reilly Media, 2013. 370 p. **7.** *Rasmusson J.* Agile Samurai: How Agile Masters Deliver Great Software [Текст] / Rasmusson, J. W.: Pragmatic Bookshelf, 2010. 237 p. **8.** *Fowler M.* Refactoring: Improving the Design of Existing Code [Текст] / Fowler, M. L.: Addison-Wesley Professional, 1999. 281 p.

Поступила в редколлегию 11.12.2015

Жмаева Юлия Владимировна, студентка каф. Искусственного интеллекта ХНУРЭ Адрес: Украина, 61166, Харьков, пр. Науки,14.

Удовенко Сергей Григорьевич, д-р техн. наук, профессор кафедры электронных вычислительных машин ХНУРЭ. Адрес: Украина, 61166, Харьков, пр. Науки,14, тел. 057-7021453, e-mail: serhii.udovenko@nure.ua.

Чалая Лариса Эрнестовна, канд. техн. наук, доцент кафедры искусственного интеллекта ХНУРЭ. Адрес: Украина, 61166, Харьков, пр. Науки, 14, тел. 057-7021337, e-mail: larysa.chala@nure.ua.