**УДК 311**

# ПРИМЕНЕНИЕ ИМИТАЦИОННОГО МОДЕЛИРОВАНИЯ ДЛЯ РАСПРЕДЕЛЕНИЯ РЕСУРСОВ В ПРОЦЕССЕ РАЗРАБОТКИ ПРОГРАММНОГО ОБЕСПЕЧЕНИЯ ПО МЕТОДОЛОГИИ SCRUM

**Смирнова С.С., Тимофеев И.О.**

*Владимирский государственный университет имени А.Г. и Н.Г. Столетовых*

*E-mail: sss_157@mail.ru, kiberaction@yandex.ru*

В настоящей статье рассмотрена методология гибкой разработки SCRUM, а также ее составляющие. Описана модель, характеризующая разработку в рамках одного спринта проекта, ее ограничения, особенности реализации. Представлена диаграмма классов модели, описаны входные данные, а также результаты многократного запуска модели. На основании полученных статистических данных сделаны выводы о составе проектной команды для решения задач различного уровня сложности. Обозначены направления развития и расширения модели.

**Ключевые слова:** SCRUM, спринт, модель, имитационное моделирование.

# THE USE OF SIMULATION FOR THE RESOURCES DISTRIBUTION IN THE PROCESS OF SOFTWARE DEVELOPMENT BY MEANS OF SCRUM METHODOLOGY

**Smirnova S.S., Timofeev I.O.**

This article deals with the agile development methodology of SCRUM and its components. The model characterizing the project development in the framework of a sprint, its limitations, peculiarities of its implementation. The class diagram of the model is presented, input data is described and the results of the model's multiple start. Conclusions about the composition of the project team to meet the challenges of various difficulty levels based on the received statistical data are made. Indicated The development and expansion directions of the model are indicated.

**Keywords:** SCRUM, sprint, model, simulation.

## Введение

Актуальным вопросом в области производства программного обеспечения является выбор методологии управления программными проектами. Относительно недавно появились так называемые «гибкие» методологии разработки ПО, которые нацелены на минимизацию рисков, путём сведения разработки к серии коротких циклов, называемых итерациями. Каждая итерация сама по себе выглядит как программный проект в миниатюре, и включает все задачи, необходимые для реализации определенной части функционала [1].

Вопрос планирования времени выполнения проекта, а также распределение задач по итерациям является одним из ключевых моментов при выборе методологии разработки ПО. Существует множество методологий гибкой разработки, такие как Extreme Programming, SCRUM, Agile Modeling, ICONIX и др. Остановимся поподробнее на методологии SCRUM, которую можно применять не только в разработке программных продуктов, но и в других видах деятельности, в которых требуется коллективная работа.

**Методология SCRUM**

SCRUM – методология гибкой разработки программного обеспечения, позволяющая правильно управлять ресурсами и максимально использовать потенциал команды. В настоящее время данная методология достигла пика своей популярности и применяется во многих ведущих IT-компаниях [2].

Для того чтобы эффективно использовать ресурсы при разработке программных продуктов, необходимо оценить, какие факторы влияют на использование данных ресурсов и как эти факторы можно изменить для получения наилучшего результата.

Таким образом, целью проведения исследования является поиск оптимального состава проектной команды для разработки программного обеспечения по методологии SCRUM.

Процесс разработки представляет собой сложную совокупность ресурсов различного рода и их взаимодействия между собой для выполнения поставленных задач. Методология SCRUM подразумевает разделение разработки проекта по спринтам - итерациям, жестко фиксированным по продолжительности, в результате которых будет реализован и передан заказчику конкретный функционал разрабатываемого продукта [3].

**Применение моделирования для описания процесса разработки по SCRUM**

Исследование процесса разработки по SCRUM в действительности является трудоемким и долговременным, поэтому, чтобы упросить экспериментальное изучение и описать этот процесс воспользуемся имитационным моделированием. Данный вид моделирования представляет собой не копирование изучаемого объекта, а имитацию процесса его функционирования в виде элементарных явлений, составляющих изучаемый процесс, с сохранением их логической структуры и последовательности протекания во времени [4].

Таким образом, необходимо построить, описать и запустить модель, характеризующую разработку в рамках одного спринта проекта.

Приведем несколько базовых понятий, которыми будем оперировать:

- Модель - имитация спринта процесса разработки ПО.
- Ресурсы - члены команды - тестировщики и разработчики.

- Задачи - мероприятия, которые необходимо выполнить в процессе спринта.

- Дефект - изъян в компоненте или системе, который может привести компонент или систему к невозможности выполнить требуемую функцию [5]. Дефект может возникнуть в результате выполнения задачи.

- Трудоемкость - количество рабочего времени, затраченного членом команды на выполнение задачи.

Ограничения модели:

- квалификации среди разработчиков и среди тестировщиков приблизительно равны;

- каждый член проектной команды в один промежуток времени может работать только над одной задачей;

- трудоемкость задач измеряется в человеко-часах, допускаются только целочисленные значения;

- шаг модельного времени равен одному человеко-часу;

- существует две очереди задач - для разработчиков и тестировщиков;

- разработчик и тестировщик может брать первую попавшуюся задачу из очереди;

- трудоемкость возникшего дефекта не может быть больше трудоемкости задачи, в рамках которой этот дефект был обнаружен

На рисунке 1 представлена диаграмма классов, которая отражает статическую структуру модели в терминологии объектно-ориентированного программирования, а также отношения между отдельными сущностями предметной области [6].

Элементы модели:

- Group - проектная команда, состоящая из разработчиков и тестировщиков, а также соответствующих очередей.

- Actor - член проектной команды, имеет количество часов в спринте и количество часов, потраченных на выполнение задач.

- Tester - тестировщик.

- Developer - разработчик.

- Task - задача, характеризуется типом, сложностью для тестировщика и сложностью для разработчика.

- TaskType - перечисление возможных типов задач.

- Runner - класс исполнения программы, содержащий основной метод запуска. Необходим для установления начальных настроек модели, считываемых из входного файла.

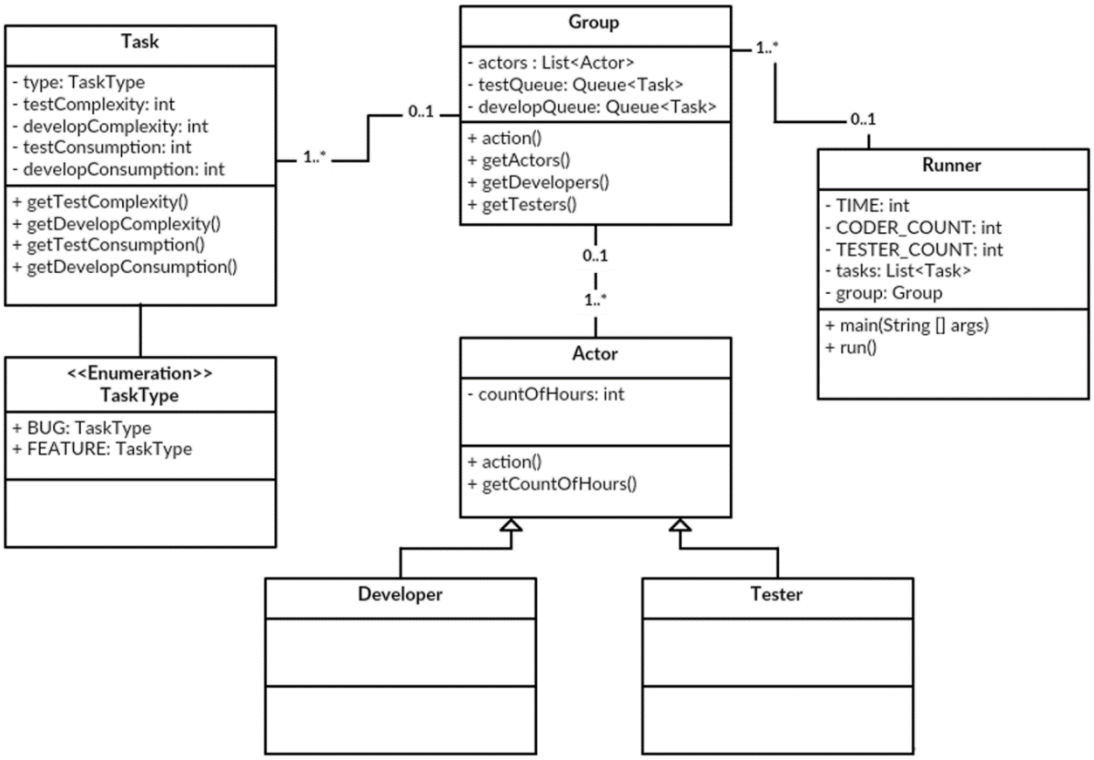


*Рисунок 1 - Диаграмма классов модели.*

**Описание работы модели**

Модель представляет собой программу на языке Java. Приведем описание работы данной модели.

На вход программе подается файл, содержащий начальные настройки модели – число разработчиков и тестировщиков (TESTER_COUNT и CODER_COUNT), продолжительность спринта (TIME), количество и трудоемкость задач для тестировщиков и разработчиков (в виде матрицы).

После загрузки данных, все задачи попадают в очередь для разработчиков (developQueue). Каждый разработчик берет на исполнение по одной задаче из очереди. По окончании исполнения задача переходит в статус “In Testing” и перемещается в очередь для тестировщиков (testQueue). Тестировщики берут из своей очереди задачу на тестирование, по окончании которой могут быть сгенерированы задачи-дефекты, уходящие в очередь для разработчиков. Вероятность возникновения дефекта для данной модели равна 0,5, т.к. вероятность того, что будет допущена ошибка, равна вероятности того, что ошибки не будет.

Все действия выполняются в рамках отведенного времени на спринт и с шагом в один человеко-час. По окончании модельного времени программа выдает информацию о наилучшем составе проектной команды, а именно, количество тестировщиков и разработчиков, а также вероятность завершения всех задач в срок.

Запустим модель 100 раз на различных вариациях параметров TESTER_COUNT и CODER_COUNT, изменяя при этом число и сложность задач. Также установим предельное число участников команды 10 человек, а модельное время 80 человеко-часов.

**Результаты запуска модели**

Соберем итоговые результаты многократного запуска модели в таблицу 1, отражающую наилучшие сочетания параметров TESTER_COUNT, CODER_COUNT. Согласно полученным данным можно сделать вывод, что наилучшим сочетанием для данного плана является сочетание, при котором команда завершит проект в запланированный срок (продолжительность спринта – 2 недели), потратив наименьшее время, при этом вероятность удачного выполнения будет наибольшей.

Таблица 1 - Итоговые результаты многократного запуска модели

| Количество и трудоемкость задач | Оптимальное число тестировщиков | Оптимальное число разработчиков | Вероятность уложиться в срок |
|---|---|---|---|
| 1 задача на 30 часов | 1 | 5 | 100% |
| 2 задачи по 15 часов | 2 | 4 | 100% |
| 3 задачи по 10 часов | 3 | 4 | 100% |
| 5 задач по 6 часов | 5 | 2 | 100% |

Проанализировав данные из таблицы 1, можно определить закономерность - чем больше задач и меньше их трудоемкость, тем больше требуется тестировщиков и меньше программистов. Это возникает в результате того, что общая трудоемкость задач одинакова, однако при работе над задачей большой трудоемкости вероятность получить более трудоемкий дефект гораздо выше и, соответственно, разработчиков необходимо больше. Число тестировщиков увеличивается ввиду увеличения числа задач, поэтому периодически появляются возможности распараллеливания тестирования.

**Заключение**

Таким образом, можно сделать вывод, что наибольшее влияние на процесс разработки программного обеспечения по методологии SCRUM оказывает высокая трудоемкость поставленных задач. Разделение таких задач на подзадачи способствует меньшему объему работ, необходимых на исправление дефектов в системе, однако требует больше ресурсов для более эффективного тестирования.

Разработанная модель охватывает лишь часть ограничений и обстоятельств, которые могут возникнуть в процессе выполнения проекта по методологии SCRUM. Чтобы провести более

сложные исследования, необходимо учитывать такие особенности как приоритет выполняемых задач, квалификацию разработчиков и тестировщиков. Для этого предполагается расширение данной модели путем имитации не только одного спринта, но и всего проекта, а также путем добавления новых ролей в команде и новых типов задач.

**Список литературы**

1. Кон М. Scrum. Гибкая разработка ПО. - Изд-во Вильямс, 2013. - 576 с.
2. Хенрик К. Scrum и XP: заметки с передовой. - Изд. дом: C4Media, 2007. - 94 с.
3. Вольфсон Б. Гибкое управление проектами и продуктами. – Изд-во Питер, 2016. - 144с.
4. Рябов О.А. Моделирование процессов и систем. Учебное пособие / Красноярск, 2008 – 122 с.
5. Стандартный глоссарий терминов, используемых в тестировании программного обеспечения Версия 2.3 (от 9 июля 2014 года) – URL: http://www.rstqb.org/fileadmin/user_upload/redaktion/rstqb_ru/downloads/ISTQB_Glossary_Russian_v2.3.pdf. Дата обращения: 15.11.2016.
6. Леоненков А.В. Самоучитель UML. - 2-е издание. - БХВ – Петербург, 2004. – 256 с.