ВЕСТНИК ТОМСКОГО ГОСУДАРСТВЕННОГО УНИВЕРСИТЕТА
2015 Управление, вычислительная техника и информатика № 3 (32)

УДК 004.04
DOI 10.17223/19988605/32/10

О.А Змеев, А.М. Политов, Я.М. Чайка

КОНЦЕПЦИЯ УНИФИЦИРОВАННОЙ МОДЕЛИ СЦЕНАРИЯ ВАРИАНТА ИСПОЛЬЗОВАНИЯ ДЛЯ ФИКСАЦИИ ФУНКЦИОНАЛЬНЫХ ТРЕБОВАНИЙ К ПРОГРАММНОМУ ПРОДУКТУ

Разработана унифицированная модель сценария варианта использования, поддерживающая различные его представления. Реализовано упрощение одновременной поддержки нескольких различных способов представления сценария варианта использования, например текстовых и графических. Предложена собственная графическая нотация модели. Построен прототип инструмента для работы с моделью, поддерживающий двустороннее преобразование в текстовую нотацию R.J. Wirfs-Brock.
Ключевые слова: объектно-ориентированный подход; управление требованиями; вариант использования; сценарий варианта использования; спецификация варианта использования; UML.

Предложенная Иваром Якобсоном (Ivar Jacobson) в конце 90-х годов прошлого века концепция определения требований к программным системам в виде вариантов использования [1] на сегодняшний день является основой рабочего потока определения требований в большинстве процессов разработки программного обеспечения, в основе которых лежит объектно-ориентированная парадигма. Согласно стандарту унифицированного языка моделирования (UML) вариант использования – спецификация множества действий, выполняемых системой, имеющего значимый результат для одного или нескольких актеров (пользователей или других систем) [2]. С помощью модели вариантов использования в современных процессах разработки фиксируются две важнейшие составляющие модели требований: структурная (в виде диаграмм вариантов использования) и динамическая (в виде спецификаций сценариев вариантов использования).

К сожалению, ориентированный прежде всего на разработку графических моделей артефактов программных систем UML предлагает для описания сценариев вариантов использования формат графических диаграмм состояний и практически не рассматривает инструменты для работы с текстовыми спецификациями сценариев вариантов использования. Отсутствие принятого, удобного для всех участников процесса разработки и заинтересованных сторон стандарта описания функциональных требований приводит к широкому разнообразию форматов текстовых спецификаций, например, в [4] рассматривается порядок сорока различных способов описания сценариев вариантов использования. Несмотря на большое число предлагаемых форматов, разнообразные представления вариантов использования можно разбить на два принципиально различных вида:

1. Текстовое представление – описывают последовательность действий на естественном языке, с той или иной степенью формализма.

2. Графическое представление – описывают последовательность действий при помощи диаграмм, как правило, имеющих формальную модель.

Необходимо отметить еще тот факт, что различные участники процесса разработки, задействованные в работе с вариантами использования, очень часто предпочитают применять разные формы представления функциональных требований. Заказчики предпочитают текстовые описания на естественном языке, поскольку для применения диаграмм необходимо знание модели и графической нотации, более того, наличие такого текстового представления является одним из достоинств подхода, основанного на применении вариантов использования. С другой стороны, разработчики предпочитают графические представления в виде диаграмм, поскольку такие представления имеют формальную модель, которая может использоваться для дальнейших преобразований в другие необходимые артефакты

процесса разработки. В результате в больших проектах со сложной структурой модели вариантов использования, когда они необходимы в различных форматах, нужно вручную создавать все необходимые представления и впоследствии выполнять синхронизацию изменений. Для больших программных систем как само такое преобразование, так и поддержка различных представлений вариантов использования в актуальном состоянии ведут к достаточно серьезным издержкам.

Для решения данной проблемы в настоящей работе предлагается применять разработанную унифицированную модель вариантов использования, которая поддерживает различные способы представления. В качестве примера поддерживаемых представлений в настоящей работе рассматриваются диаграммы деятельности UML (Activity Diagram) [2], текстовые спецификации, которые разработал A. Cockburn [3] и двухколоночный табличный формат текстовых спецификаций, разработанный R. Wirfs-Brock [5].

1. Представления сценариев варианта использования

Для классификации различных определений вариантов использования можно применять следующие критерии, предлагаемые Alistair Cockburn в [3]:

1. Цель: является ли целью сбор пользовательских историй или описание требований.

2. Содержимое: является ли содержимое варианта использования согласованным или оно может быть противоречивым. Если содержимое согласованно, то является ли оно при этом формальным.

3. Множественность: является ли вариант использования представлением одного или множества сценариев.

4. Структура: образует ли множество вариантов использования формальную (или полуформальную) структуру.

И Якобсон [1], и Коберн [3] определяют вариант использования согласно данной классификации как требование, имеющее согласованное содержимое, представляющее множество сценариев и образующее полуформальную структуру.

Для представления структуры варианта использования Коберн [1] предложил модель «полосатых брюк» (рис. 1). «Ремень брюк» – цель варианта использования, которая «держит» все сценарии. «Полосы» – индивидуальные сценарии, характеризуемые условием их наступления и результатом. Множество «Полос» разделено на две группы: достигающие цели и не достигающие цели. Основным сценарием в данной модели является наиболее простой, в котором все промежуточные цели достигаются. Все остальные сценарии являются альтернативными. Альтернативные сценарии делятся на восстанавливаемые (если одна или несколько промежуточных целей не были достигнуты, но итоговая цель была достигнута) и неудачные (итоговая цель не была достигнута).

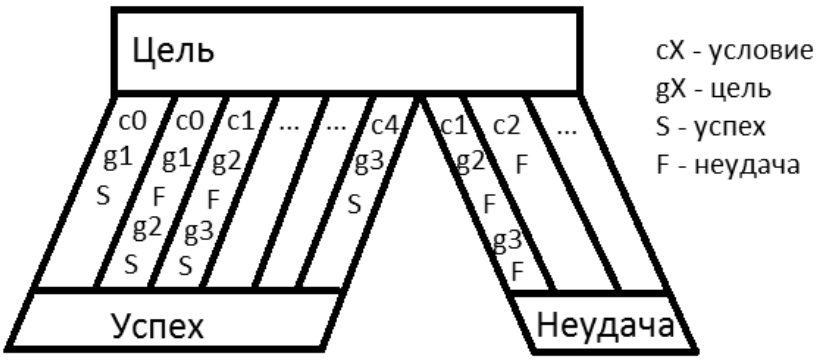

Рис. 1. Модель «полосатых брюк»

Как уже упоминалось выше, в процессе разработки программного обеспечения варианты использования применяются (могут использоваться) для фиксирования функциональных требований к разрабатываемому продукту. Часто при этом их необходимо представлять заказчику (или иным заинтересо-

ванным лицам, не занимающимся напрямую разработкой), который может не знать специализированных нотаций (например, UML). В таком случае единственным способом представления варианта использования для заказчика является текстовое описание на естественном языке, для применения которого не требуется знаний специализированных нотаций.

С другой стороны, на основании вариантов использования происходят дальнейшие этапы разработки программного обеспечения (анализ, проектирование, тестирования). В таком случае для возможности автоматизированной обработки вариантов использования необходима формальная модель, которая, как правило, отсутствует у текстовых представлений, но присутствует у графических.

Таким образом, при необходимости применения различных представлений варианта использования возможны следующие варианты:

1. Вариант использования специфицируется в различных его представлениях вручную. В таком случае спецификатору ВИ необходимо проделывать фактически одну и ту же работу (спецификацию одного и того же ВИ) несколько раз. В случае необходимости внесения изменений в вариант использования такие изменения должны быть отражены во всех его представлениях, что также добавляет дополнительную работу и может являться причиной возникновения ошибок (из-за расхождений между представлениями).

2. Вариант использования специфицируется в одном представлении. После этого, с помощью специализированных инструментов, выполняется его преобразование (в том числе и автоматическое) в необходимые представлениях. При внесении изменений они вносятся в первичное представление, что ограничивает возможность спецификатора вариантов использования редактировать производные представления (после повторного преобразования будут теряться изменения производных представлений вариантов использования, изменения в производных представлениях вариантов использования не будут отражены в первичном). Такая модель реализован, например, в Visual Paradigm [6].

3. Решение, которое предлагается в настоящей работе: создание унифицированной модели вариантов использования, которая поддерживала бы необходимые форматы представления вариантов использования и не ограничивала редактирование варианта использования одной моделью (рис. 2).

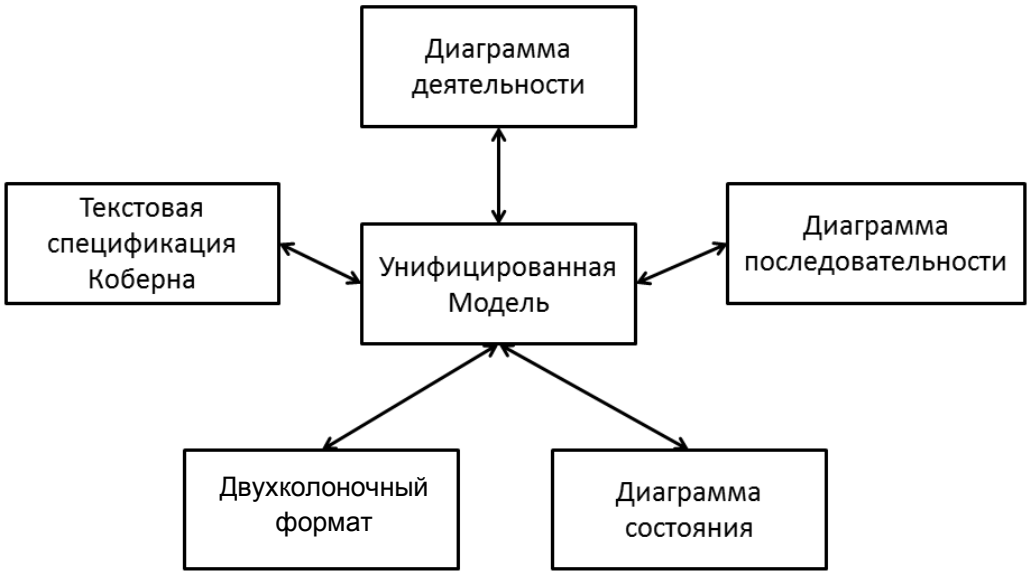

Рис. 2. Переход между представлениями с помощью унифицированной модели

2. Представления сценариев варианта использования

Существует достаточно много способов представления вариантов использования [4]. В качестве примера в данной работе будут рассмотрены три из них.

2.1. Диаграммы деятельности UML

Одним из способов графического представления вариантов использования являются диаграммы деятельности UML. Данный вид диаграмм позволяет определить поведение с помощью последовательного исполнения поведений более низкого уровня. Исполнение следующего действия может начинаться в результате наступления одного из следующих событий: завершение исполнения предыдущего действия (потоки управления), появление необходимых данных (потоки данных), наступление определенного события (потоки управления).

2.2. Текстовая спецификация А. Коберна

Согласно [1] текстовая спецификация вариантов использования состоит из общей информации: названия варианта использования, контекста использования, области действия, уровня цели, основных актеров, предусловия, триггера, минимальных гарантий и успешного постусловия.

Кроме общей информации о варианте использования, описываются его основной сценарий и расширения к нему. Основной сценарий представляет собой последовательность действий, при успешном выполнении которых достигается цель варианта использования. Расширения основного сценария описывают действия при возникновении исключительных ситуаций (действия, не предусмотренные основным сценарием, ошибки, внешние события).

Основной сценарий описывается в виде пронумерованной последовательности шагов:

<номер шага>. <действие>

Расширения привязываются к определенным шагам основного сценария (в том числе ко всему сценарию) и описываются в виде пронумерованной последовательности шагов:

<номер шага + идентификатор расширения>. <условие>: <действие или вложенный вариант использования>

2.3. Двухколоночный формат Р. Вирфс-Брок

В [5] Ребеккой Вирфс-Брок было предложено оформлять сценарии варианта использования в виде диалога между основным актером и системой (не предполагается участие в одном варианте использования более двух актеров). Сценарий записывается в форме таблицы, состоящей из двух колонок: действия основного актера и действия системы.

3. Основа унифицированной модели

В результате анализа различных представлений вариантов использования были выделены следующие особенности:

1. Вариант использования имеет множество сценариев, завершающихся успешно или неудачно.
2. Каждый такой сценарий представляет собой последовательность действий.
3. Каждое действие исполняется определенным актером (например, пользователем, системой).
4. Множество таких сценариев может быть бесконечно (из-за возможности образования циклов) и в результате неприменимо в чистом виде на практике. Для решения данной проблемы выделим основной сценарий варианта использования (сценарий, все действия которого успешно выполняются), а остальное множество сценариев определим через расширения к отдельным действиям основного сценария (в том числе через расширения ко множеству действий и, возможно, ко всему сценарию).

4. Предлагаемая модель сценария

Предлагаемая модель представляет собой расширение к модели UML (версии 2.4.1, стандартизованной ISO) [2]. Модель основана на пакетах Classes::Kernel и CommonBehaviors::BasicBehaviors. На рис. 3 показана диаграмма пакетов предлагаемой модели.

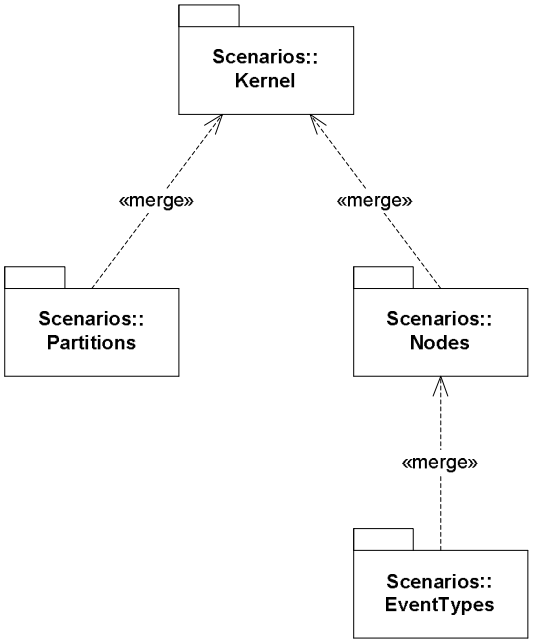

Рис. 3. Диаграмма пакетов предлагаемой модели

Модель состоит из следующих пакетов:

1. Scenarios::Kernel содержит ядро модели, определяющее базовые элементы сценария и потоки управления для перехода между элементами.

2. Scenarios::Nodes содержит конкретные элементы сценария (например, действия, обработчики событий и т.д.).

3. Scenarios::Partitions содержит разделы, используемые для группировки элементов сценария.

4. Scenarios::EventTypes содержит типы событий.

Рассмотрим каждый пакет более подробно.

4.1. Пакет Scenarios::Kernel

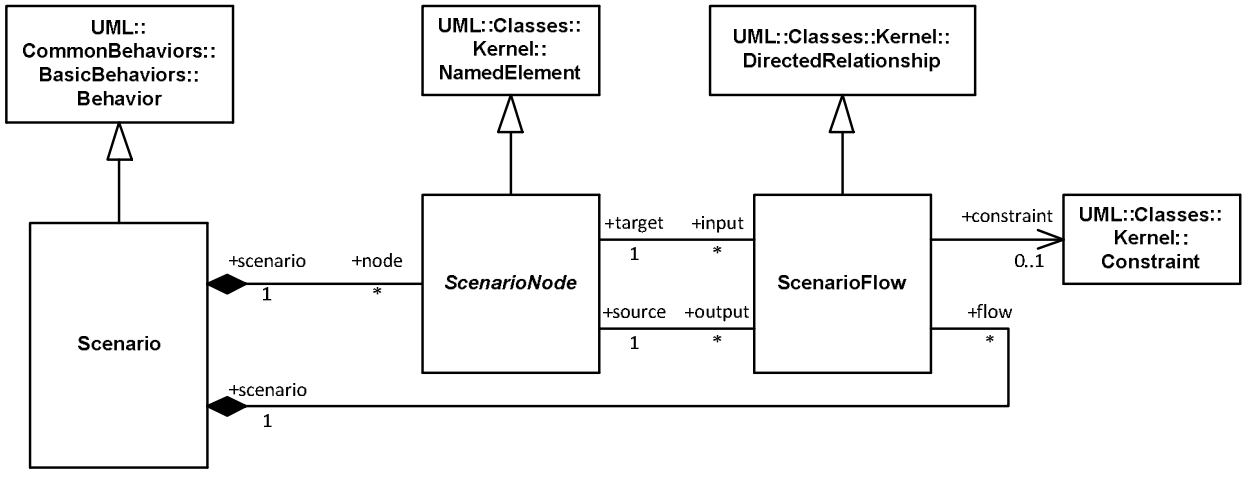

Рис. 4. Диаграмма классов пакета Scenarios::Kernel

Пакет Scenario::Kernel представляет собой ядро унифицированной модели (рис. 4). В нем определяются сценарий варианта использования, элементы сценария и потоки управления, используемые для перехода между элементами сценария.

Описание классов пакета:

1. Scenario представляет собой сценарий варианта использования.

2. ScenarioNode представляет собой элемент сценария варианта использования.

3. ScenarioFlow представляет направленный поток управления, передающий исполнение от одного элемента сценария другому. Поток может иметь ассоциированное с ним условие, в таком случае передача управления выполняется, только если условие выполняется.

4.2. Пакет Scenarios::Nodes

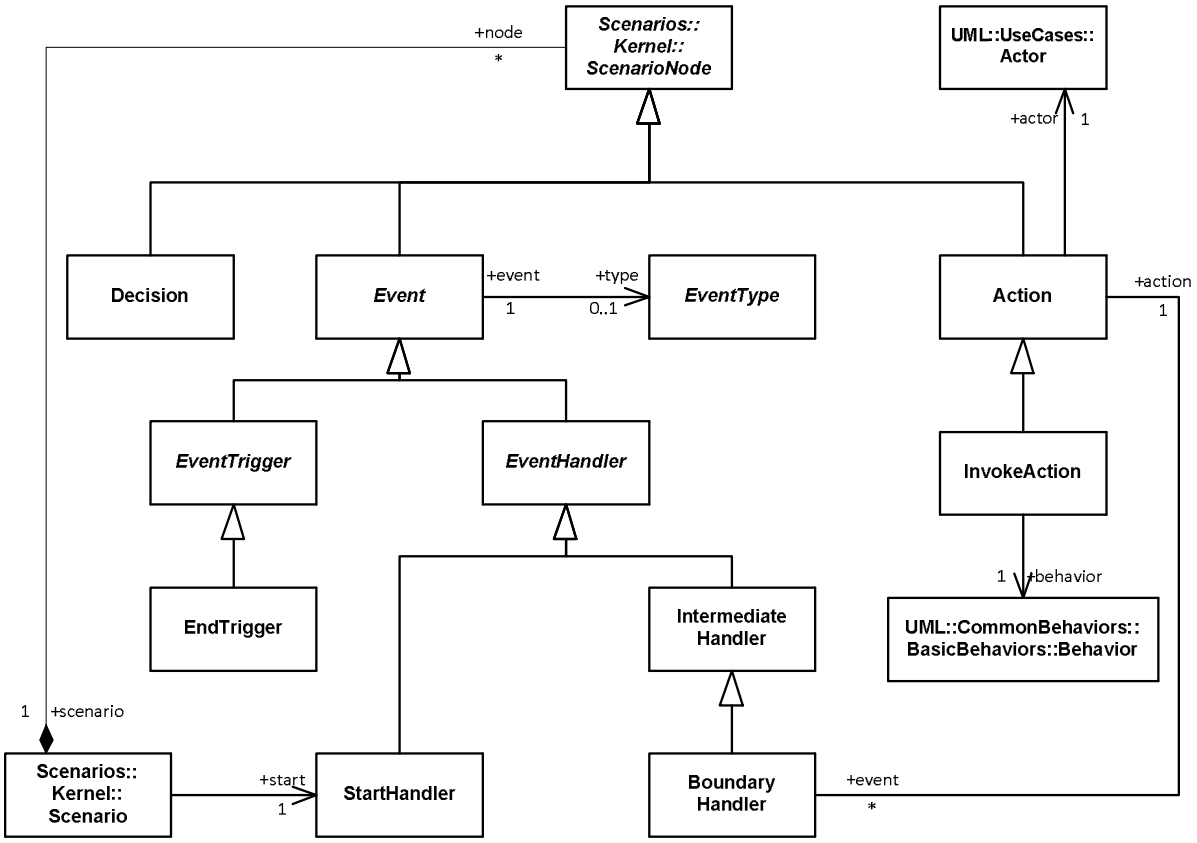

Рис. 5. Диаграмма классов пакета Scenarios::Nodes

Пакет Scenarios::Nodes содержит различные элементы сценария унифицированной модели (рис. 5). В нем определяются действия сценария, триггеры и обработчики событий и т.д.

Описание классов пакета:

1. Scenario – класс сценария расширен для связи с обработчиком начала сценария.

2. Action представляет действие в сценарии.

3. InvokeAction представляет действие сценария, являющееся вызовом другого поведения (в том числе, возможно, и сценария).

4. Decision представляет элемент сценария, который выбирает один из исходящих потоков для продолжения исполнения. Для определения потока, по которому передается управление, используются условия потоков.

5. Event – абстрактный класс, представляющий событие в сценарии: элементы, вызывающие события, и элементы, обрабатывающие события.

6. EventType – абстрактный класс EventType, представляет тип события.

7. EventTrigger – абстрактный класс EventTrigger, представляет триггер события – элемент, при исполнении которого происходит определенное событие.

8. EndTrigger – класс EndTrigger, представляет завершающий сценарий элемент. При исполнении данного элемента сценарий завершается. Если с завершающим элементом ассоциирован тип события, то происходят вызов данного события и передача его в сценарий, вызвавший данный сценарий.

9. EventHandler – класс EventHandler, представляет абстрактный обработчик события.

10. StartHandler – класс StartHandler, представляет обработчик начала сценария – элемент сценария, с которого начинается его исполнение. У сценария может быть только один обработчик начала.

11. IntermediateHandler – класс IntermediateHandler, представляет промежуточный обработчик события.

12. BoundaryHandler – класс BoundaryHandler, представляет промежуточный обработчик события, который обрабатывает события в связанном с ним действии сценария.

Заключение

В результате была разработана унифицированная модель вариантов использования, имеющая собственную графическую нотацию.

На основе предложенной модели разработан прототип инструмента для работы со сценариями варианта использования в предложенной модели. Реализация поддерживает преобразование сценариев из текстовой нотации Ребекки Вирфс-Брок (R. J. Wirfs-Brock) в графическую нотацию предложенной модели и наоборот. В дальнейшем планируется анализ возможности трассировки вариантов использования в предложенной модели в артефакты дальнейших этапов процесса разработки, например модель анализа и модель тестирования.

ЛИТЕРАТУРА

1. *Jacobson I.* Modeling with Use Cases: Formalizing use-case modeling // JOOP. 1995. June. V. 8, No. 3.
2. ISO/IEC 19505-2:2012(E). Information technology – Object Management Group Unified Modeling Language (OMG UML). Part 2: Superstructure. April 2012. International Organization for Standardization, 2012. 740 p.
3. *Cockburn A.* Structuring use cases with goals. Electronic version of printed publication. 1995. URL: http://alistair.cockburn.us/Structuring+use+cases+with+goals (access date: 30.11.2011).
4. *Hurlbut R.R.* A Survey of Approaches For Describing and Formalizing Use Cases. Expertech, Wheaton, Illinois, 1997. URL: http://www.iit.edu/~rhurlbut/xpt-tr-97-03.html (access: 14.12.2011).
5. *Wirfs-Brock R.* Designing Scenarios: Making the Case for a Use Case Framework // The Smalltalk Report. 1993. November-December. V. 3, No. 3.
6. *Software* Design Tools for Agile Teams with UML, BPMN and More. URL: http://www.visual-paradigm.com (access date: 23.03.2014).

Змеев Олег Алексеевич, д-р физ.-мат. наук, профессор. E-mail: ozmeyev@gmail.com
Политов Арсентий Михайлович. E-mail: m.politov@gmail.com
Чайка Яна Михайловна. E-mail: chaykayana@gmail.com
Томский государственный университет

Поступила в редакцию 26 мая 2015 г.

Zmeyev Oleg A., Politov Arsenty M., Chayka Yana M. (Tomsk State University, Russian Federation).
Concept of unified use case model for software requirements specification.
Keywords: object-oriented approach; requirements management; use case, use case scenario; use case specification; UML

DOI 10.17223/19988605/32/10

Use case is a specification of actions sequence executed by system for achieving a significant result for any or all actors. The use cases model is used for capturing both structural and behavioral parts of product's functional requirements in current developing processes.

Unified Modelling Language offers several graphical representations of use cases scenarios, such as activity diagrams, rather than textual formats. Nevertheless, there are many different textual formats used for use cases scenarios specifications, e.g. Wirfs-Brock's two-column table style and Cockburn style. Different stakeholders may prefer a different specification format, e.g. customer may prefer textual formats, while developers may prefer graphical formats. In addition, use cases can be used for subsequent development stages (e.g. analysis, design, testing). For use cases to be automatically traced into these stages they must be formally defined. Such definition is usually possible in the graphical representation but not in textual.

When several formats are required, following approaches can be applied:

1. Use case should be created manually in every required formats and every change should be to be synchronized between used formats to keep requirements specifications up to date. Such synchronization is time consuming and error-

prone process due to the human factor. To simplify this process, the development of the unified use cases model that will support different formats of use cases scenarios is proposed.

2. Use case is specified in a single format. Then transformation into required formats is performed by special tools. Any changes in the use case are made in the primary format of use case thus limiting an ability to edit derived formats (subsequent transformation will cause a loss of derived formats edits and these edits will not affect primary format).

3. An approach proposed in this paper is to create the unified use case model that will support different formats of use cases scenarios and not restricting scenario editing to one of them.

There are the following use cases features:

1. Use case has multiple scenarios that can either succeed or fail.

2. Each scenario is described by actions sequence.

3. Each action is executed by a specific actor.

4. A set of these scenarios can be infinite due to loops and cannot be applied. To make scenarios more useful, a main success scenario can be selected and other scenarios can be defined as extensions to specific actions of a main success scenario.

The model proposed in this paper is an extension of UML 2.4.1 (formally published as ISO/IEC 19505-1 and ISO/IEC 19505-2). It is based on the following packages: Classes::Kernel and CommonBehaviors::BasicBehaviors and consists of the following:

1. Scenarios::Kernel package that contains the unified model kernel. In this package use case scenarios, scenario nodes and flows are defined.

2. Scenarios::Nodes package that contains specific scenario nodes, such as scenario actions, triggers, handlers, decisions, etc.

3. Scenarios::Partitions package that defines partitions used for scenario nodes grouping.

4. Scenarios::EventTypes package that define events types, e.g. generic errors, system events, user events, time events, etc.

As a result, the unified use case model that has its own graphical notation is developed. In addition, prototype of a tool that supports editing use cases in the developed model and transformation between Wirfs-Brock's textual format and model's graphical notation is implemented.

REFERENCES

1. Jacobson, I. (1995) Modeling with Use Cases: Formalizing use-case modeling. *JOOP*. 8 (3).
2. ISO/IEC 19505-2:2012(E). (2012) Information *technology. Object Management Group Unified Modeling Language (OMG UML) Part 2: Superstructure*. April 2012. International Organization for Standardization.
3. Cockburn, A. (2011) *Structuring use cases with goals*. [Online] Available from: http://alistair.cockburn.us/Structuring+use+cases+with+goals. (Accessed: 30th November 2011).
4. Hurlbut, R.R. (1997) *A Survey of Approaches For Describing and Formalizing Use Cases*. Expertech, Wheaton, Illinois. Available from: http://www.iit.edu/~rhurlbut/xpt-tr-97-03.html. (Accessed: 14th December 2011)
5. Wirfs-Brock, R. (1993) *Designing Scenarios: Making the Case for a Use Case Framework*. The Smalltalk Report. 3(3).
6. *Software Design Tools for Agile Teams with UML, BPMN and More*. [Online] Available from: http://www.visual-paradigm.com. (Accessed: 23th March 2014).