УДК 004.91

И. С. Печенко, науч. сотр., e-mail: ivan.pechenko@intel.com,
О. В. Венгер, ст. науч. сотр., e-mail: oleg.v.venger@intel.con,
ЗАО "Интел А/О"

Способ автоматизированного создания диаграмм последовательности операций для сценариев поведения системы, описанных в виде текста

Одним из способов спецификации поведения вычислительных систем является описание набора сценариев, по которым должно определяться взаимодействие ее компонентов. 95 % такого рода спецификаций составляют документы, в которых сценарии описаны в виде текста. Такому описанию присущи неточности, неоднозначности, неполнота, невозможность машинной обработки. Предлагается способ преобразования текстовой спецификации к виду, пригодному для машинной обработки и более подходящему для обмена информацией между группами участников проекта. В текстовое описание добавляются тэги, позволяющие структурировать спецификацию. Затем, с использованием графической нотации BPMN, из структурированного текста автоматически создается диаграмма последовательности операций. Предлагаемый способ был реализован и протестирован на нескольких моделях сценариев поведения систем. Данный подход может быть применен на практике для сценариев, описывающих поведение сложных вычислительных систем и комплексов.

Ключевые слова: *модель поведения, сценарий поведения, структурированный текст, машинная обработка спецификации, диаграмма последовательности операций, BPMN*

Введение

Современные вычислительные системы состоят из большого числа различных программных и аппаратных компонентов. Аппаратная часть насчитывает десятки блоков, таких как процессоры, графические ядра, контроллеры памяти, контроллеры устройств ввода-вывода и т. д. Программная часть включает в себя прошивки для аппаратных блоков, драйверы, операционную систему и др. Одной из задач, возникающих при проектировании таких систем, является спецификация и описание того, каким образом компоненты должны взаимодействовать между собой и с окружением вычислительной системы. Качество такой спецификации является определяющим фактором для успеха других этапов проектирования, в частности, эффективности тестирования системы. Кроме того, эффективность процесса проектирования можно повысить путем использования результатов машинной обработки спецификации в дальнейшей разработке и тестировании.

Одним из способов спецификации поведения системы является описание набора сценариев, по которым должно осуществляться взаимодействие ее компонентов. В настоящее время не существует единого стандарта для описания сценариев. Среди используемых методов можно отметить диаграммы коммуникации и последовательности UML (Unified Modeling Language, стандарт графического описания для объектного моделирования в области разработки программного обеспечения [15]), диаграммы последовательности сообщений, диаграммы последовательности операций и т. д. На практике значительную часть такого рода спецификации составляют документы, в которых сценарии описаны в виде обычного текста. Согласно [1] 95 % технических требований при разработке программного обеспечения представлены в виде обычного либо структурированного текста.

Текстовое описание сценариев является удобным начальным способом описания поведения системы. Это особенно справедливо для сложных сценариев с большим числом событий и взаимодействующих компонентов. Его легко редактировать и дополнять, в том числе вставлять новые блоки информации в произвольное место сценария. Однако описание в виде обычного текста имеет ряд недостатков. Во-первых, свободному текстовому описанию присущи неточности, неоднозначности, неполнота. Во-вторых, форма, удобная для создания спецификации, не всегда является удобной для понимания и использования ее другими участниками проекта. Для понимания больше подходит визуальное представление, например, такое как диаграмма последовательности операций. В-третьих, автоматическая обработка текстовых описаний невозможна без специальной подготовки.

Существующие работы, связанные с анализом и использованием текстовой спецификации, в боль-

шинстве случаев нацелены на улучшение процесса разработки программного обеспечения. Их можно разделить на две категории. Часть работ предлагает способы обнаружения проблем в текстовой спецификации [2, 7], оценки и улучшения качества текстовой спецификации [3—6]. Другая группа работ предлагает способы экстракции UML-диаграмм из текстовой спецификации [8—10, 13]. В обоих случаях исходное текстовое описание сначала структурируется, что обеспечивает возможность последующей машинной обработки. За счет такого структурирования достигается соответствие текста некой формализованной структуре, которую затем нетрудно перести в диаграмы UML. Такое структурирование достигается двумя основными путями: путем лингвистического анализа с последующим устранением выявленных неточностей и дефектов в текстовых спецификациях и путем написания спецификаций по заранее установленному структурированному шаблону. Интересным способом структурирования текстового описания является SBVR (Semantics of Business Vocabulary and Rules) [14], продвигаемый консорциумом OMG (Object Management Group) [12], который занимается разработкой и продвижением объектно-ориентированных технологий и стандартов. Членами консорциума являются сотни ведущих компаний-производителей программного обеспечения.

В данной работе предлагается способ преобразования спецификации сценариев в виде обычного текста к виду, пригодному для машинной обработки и более подходящему для обмена информацией между различными группами участников проекта, ответственными за проектирование, разработку, тестирование программно-аппаратных платформ. Суть предлагаемого решения заключается в следующем: сначала в исходное текстовое описание добавляются специальные тэги, позволяющие структурировать полученную спецификацию. Затем из текста с тэгами автоматически создается графическое представление — диаграмма последовательности операций. В отличие от методов структурирования текста спецификаций, основанных на лингвистическом анализе, описанный в данной статье метод не предполагает привязки к определенной области применения и использования словарей понятий и ключевых фраз, а преимущество описанного метода перед использованием заранее определенных шаблонов текста заключается в возможности работать с готовыми текстами спецификаций без приложения значительных усилий на их полную переработку. Предлагаемый способ был реализован и используется для преобразования реальных сценариев, состоящих из десятков шагов каждый.

Пример-иллюстрация: исходный вид

В качестве иллюстрации будем использовать описание процесса подготовки статьи для публикации в материалах научной конференции. В текстовом виде оно имеет следующий вид:

Организаторы конференции рассылают информационное сообщение. Получив сообщение, авторы готовят начальную версию статьи и отправляют ее организаторам. Организаторы рассылают полученные работы рецензентам. Рецензенты изучают присланные работы и готовят рецензии. Получив результаты рецензирования, организационный комитет принимает решение о приеме работы к публикации и информирует авторов о своем решении. В случае положительного решения авторы с учетом замечаний готовят окончательную версию статьи и слайды доклада и отправляют организаторам для корректурной правки. В случае наличия замечаний авторы должны их устранить. Вместе с этим авторы должны предоставить дополнительные документы (акт приема-передачи, лицензионное соглашение). Организаторы осуществляют проверку дополнительных документов. Процесс заканчивается после устранения замечаний.

Данное описание (хотя и неформализованное) дает представление о том, как выглядит процесс подачи статьи для публикации. Мы видим, что в процессе принимают участие три объекта (организаторы, авторы, рецензенты), мы видим, с чего начинается и чем заканчивается процесс, а также получаем представление о последовательности действий. Мы также видим, что некоторые действия происходят всегда, в то время как другие происходят только при выполнении некоторых условий. Мы получаем представление о том, как взаимодействуют различные объекты: они обмениваются сообщениями, содержащими определенные данные. Такой способ взаимодействия как раз характерен для компонентов программно-аппаратных платформ, что и позволяет использовать этот абстрактный пример для описания предложенного способа.

Структурирование примера-иллюстрации

Человек, прочитавший представленное описание, может создать на ее основе эквивалентную блок-схему, однако автоматизация создания блок-схемы затруднительна. В данной главе описывается, каким образом можно с минимальными усилиями сделать подобное описание пригодным для машинной обработки.

Выявим сущности, используемые при описании нашего сценария. Прежде всего, мы имеем объекты, осуществляющие действия. Будем называть такие объекты *агентами*. Агентами в нашем сценарии являются "Организатор", "Автор", "Рецензент". *Действие* само по себе также является сущностью. Каждое предложение исходного описания описывает какое-либо действие. Как правило, сценарий предполагает определенную последовательность действий. Некоторые действия могут осуществляться только после того, как выполнены другие действия. В тексте последовательность возникает естествен-

ным образом: действия-предшественники упоминаются раньше, чем действия-следствия. *Сообщение* — еще одна сущность, характеризующая взаимодействие между агентами. Частью сообщения являются данные, необходимые для выполнения дальнейших действий. Наконец, мы видим, что переход от действия к действию характеризуется *условием перехода*.

Первым шагом структуризации является использование нумерованного списка для указания последовательности действий. Элемент списка соответствует атомарному действию. На усмотрение автора спецификации остается гранулярность атомарных действий. Одна и та же операция может быть представлена как одно действие или как последовательность нескольких действий. В зависимости от цели использования оптимальным может являться как один, так и другой вариант. Заметим, что последовательность элементов списка не дает полного представления о сценарии в случае наличия ветвлений или параллельного выполнения действий.

Вторым шагом является однозначное определение агента, осуществляющего действие. Как правило, идентификатор агента уже присутствует: это подлежащее предложения, которое описывает действие. Для сценариев с ветвлениями или параллельными действиями иногда необходимо идентифицировать отдельные действия (как это делать, будет описано ниже).

Третьим шагом является определение всех непосредственных переходов между действиями, включая определение условий перехода. Значительная их часть не требует дополнительных действий — переход от какого-либо элемента списка к следующему элементу является переходом по умолчанию.

Вот каким образом выглядит структурированное описание нашего сценария:

*1. [[**flow**:подготовка_публикации]].*

*2. [[**start**]].*

3. [[Организатор]] Рассылает информационное сообщение.

4. [[Автор]] Готовит начальную версию статьи.

5. [[Организатор]] Расслылает полученные работы рецензентам.

6. [[Рецензент]] Изучает присланные работы и готовит рецензии.

7. [[Организатор]] Принимает решение о принятии работы к публикации.

*8. [[Автор]] Работа принята? [[**to** работа_принята, подготовка_документов 'да']] [[**to end** 'нет']],*

9. [[Автор: работа_принята]] Готовит окончательную версию статьи и слайды доклада с учетом замечаний.

10. [[Организатор]] Осуществляет корректурную правку.

*11. [[Организатор]] Есть замечания?[[**to** Автор: работа_принята 'да']] ?[[**to** конец 'нет']].*

12. [[Автор: подготовка_документов]] Предоставляет дополнительные документы. {{ИТ: сведения об авторах, экспертное заключение о возможности публикации}}.

13. [[Организатор]] Осуществляет проверку документов.

*14. [[Организатор]] Есть замечания? [[**to** Автор: подготовка_документов 'да']] [[**to** конец 'нет']].*

*15. [[Организатор:конец]] [[**merge**]].*

*16. [[**end**]].*

Структурированное описание представляет собой нумерованный список и содержит служебную информацию (тэги) внутри двойных квадратных скобок. В данном примере ключевые слова выделены жирным шрифтом для улучшения восприятия.

Текст спецификации начинается с элемента списка, содержащего тэг [[**flow**:label]], где label — имя сценария, а заканчивается элементом списка, являющегося тэгом [[**end**]]. Каждый элемент списка представляет собой задачу, выполняемую на данном этапе, или блок ветвления. Каждый элемент списка должен начинаться с тэга [[agent_name<:label>]], где agent_name — имя агента, выполняющего данное действие, а label — необязательная метка, которую можно использовать как указатель на действие или блок ветвления. Также в элементе списка могут присутствовать другие тэги:

- тэг [[**start**]] обозначает одну из возможных точек, с которых начинается выполнение сценария. Допустимо наличие нескольких стартовых точек;
- тэг [[**end**]] после спецификации задачи обозначает, что после данной задачи выполнение сценария заканивается. В отличие от предыдущего, данный тэг должен входить в состав элемента списка, описывающего задачу;
- тэг [[**to** <agent:>label <"condition">]] означает переход, отличный от перехода по умолчанию. Здесь agent — имя агента, которому передается управление, label — метка, к элементу с которой осуществляется переход, condition — условие, при котором выполняется данный переход. В случае отсутствия идентификатора агента имеется в виду переход к другому действию текущего агента. Если тэга [[**to** ...]] в элементе списка нет, то по умолчанию считается, что после выполнения задачи выполняется следующая за ней в списке. Если же этот тэг присутствует, то возможны несколько вариантов. Если такой тэг один в элементе списка, тогда после выполнения задачи, соответствующей этому элементу, осуществляется переход к элементу с указанной меткой. Если данных тэгов в элементе списка несколько, то по умолчанию считается, что данный элемент соответствует блоку, в котором происходит разветвление сценария в зависимости от условий, заданных в conditon.

Описанные выше тэги позволяют описать, как выглядит граф сценария, где вершинами являются действия, а ребрами — переходы между ними. В дополнение к этой базовой информации структурированный текст содержит дополнительные аннота-

ции внутри блоков {{...}}. Простейший случай использования аннотаций — добавление комментариев, распознаваемых при машинной обработке. Однако внутри аннотации могут определяться существенные свойства, расширяющие базовую модель сценария в контексте конкретного применения. Для того чтобы различить аннотации, добавленные для разных приложений, текст внутри блока {{...}} начинается с метки. Так, хотя в нашем примере сценарий описывает процесс подачи статьи в общем, метка ИТ говорит нам о том, что информация о дополнительных документах относится к требованиям журнала "Информационные технологии".

Преобразование структурированного текста в диаграмму

Программа для преобразования структурированного текстового описания сценариев была реализована как расширение к среде Microsoft Office. Текстовый процессор Microsoft Word был использован для спецификации сценариев. Объектная модель MS Word представляет богатый API для работы со списками, встречающимися в документе. Для визуализации сценариев был использован графический редактор Microsoft Visio. Одной из причин такого решения явилось то, что MS Visio включает в себя библиотеку графической нотации BPMN.

Как и SBVR, нотация BPMN (Business Process Model and Notation) [11] была разработана консорциумом OMG. Целью BPMN является предоставление способа описания, легко понятного широкой аудитории — от бизнес-аналитиков, создающих начальное черновое описание процессов, до разработчиков реальных технологий, реализующих эти процессы, и пользователей этих технологий.

Нотация BPMN призвана служить связующим звеном между фазой проектирования бизнес-процесса и фазой его реализации. Как указано выше, существует потребность в аналогичном связующем звене между фазами проектирования, разработки и тестирования вычислительных систем. Для описания сценариев поведения вычислительных систем мы предлагаем использовать подмножество элементов BPMN.

Для иллюстрации использования BPMN рассмотрим диаграмму примера-иллюстрации (рис. 1). Сущности, описанные в предыдущей главе, отображаются на элементы BPMN следующим образом.

A. Агенты

Агент соответствует элементу BPMN "дорожка". Создание диаграммы сценария начинается с определения набора агентов. Объекты агентов служат контейнерами для других элементов диаграммы. В нашем примере в сценарии участвуют три агента ("Организатор", "Автор", "Рецензент"). Мы используем вертикальную ориентацию агентов, поскольку она является основной для диаграмм последовательности сообщений, которые часто используются при проектировании вычислительных систем.

B. Действия

Действия описывают атомарные операции, выполняемые агентами. В нашем примере большинство действий совершает Организатор: он рассылает информационное сообщение, затем рассылает полученные работы для рецензии, принимает решение о принятии статьи к публикации и т. д. Действия соответствуют элементу BPMN "действие" и изображаются прямоугольниками со скругленными углами. Текст внутри прямоугольника содержит описание действия.

C. Блоки ветвления

Блоки ветвления описывают логические разветвления в сценарии, при которых то, какая задача будет выполняться следующей, зависит от некоего логического условия, определенного внутри него. Этот элемент соответствует элементу BPMN "шлюз" и изображается ромбом. В нашем примере сценарий будет иметь разный вид для принятых и отклоненных работ, что и указано наличием блока ветвления внутри агента "Автор".

Разновидностью блока ветвления является элемент объединения, соответствующий элементу "параллельный шлюз" BPMN. Он изображается ромбом, внутри которого помещен знак '+'. Наличие элемента объединения перед действием указывает на то, что агент начинает выполнять действие только после того, как завершатся все действия-предшественники. В случае отсутствия элемента объединения перед действием предполагается, что агент начинает действие, когда хотя бы одно из действий-предшественников завершено.

D. Соединяющие объекты (коннекторы)

Коннекторы определяют последовательность выполнения задач. Коннектор типа "последовательность" соединяет задачи внутри одного объекта и соответствует элементу BPMN "поток управления" и изображается сплошной линией со стрелкой, указывающей направление перехода. Коннектор типа "сообщение" соединяет задачи в разных агентах и таким образом определяет, как происходит взаимодействие агентов. Он изображается штриховой линией. Сообщения могут представлять, например, команду одного агента другому, отклик на ранее полученную команду и т. п. Такое использование сообщений позволяет, помимо прочего, сократить структурированное текстовое описание. Свободный текст *"Организаторы конференции рассылают информационное сообщение. Получив сообщение, авторы готовят начальную версию статьи ...."* превращается в лаконичное:

1. [[Организатор]] рассылает информационное сообщение.

2. [[Автор]] готовит начальную версию статьи.

Наличие в нумерованном списке смежных элементов с разными идентификаторами агентов подразумевает пересылку сообщения от первого ко второму. С коннекторами может быть связано условие перехода от задачи к задаче, когда задача-следствие активируется только при выполнении определенного условия. Условие изображается как текст, ассоциированный с коннектором. "Сообщение" соответствует элементу BPMN "поток сообщений".

E. Аннотации

Аннотации, добавляемые к объектам диаграммы, служат для уточнения значения элементов диаграммы и повышения ее информативности. Для изображения аннотаций на диаграмме-примере используются объекты-ссылки с текстом, содержащимся внутри блока {{...}}.

F. Вспомогательные элементы

Нотация BPMN также содержит элементы "начало" и "конец". Они используются для улучшения читаемости диаграммы. Для обозначения начала сценария используется символ окружности. Для обозначеня конца сценария используется также символ окружности жирной линией. В контексте задачи спецификации поведения вычислительных систем эти элементы применяются для обозначения действий, с которых может начинаться сценарий, и действий, после которых сценарий завершается. В общем случае сценарии может иметь несколько элементов "начало" и "конец".

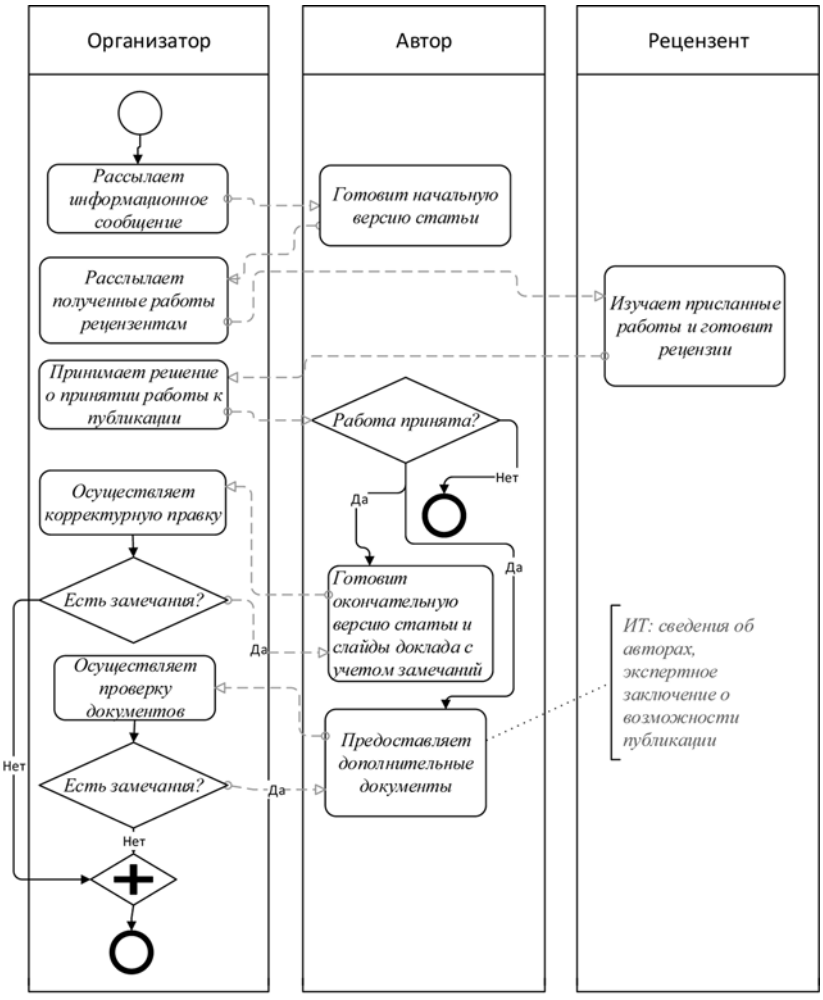

Рис. 1. Процесс подготовки статьи к публикации

Реализация генератора диаграмм

Диаграмма на рис. 1 была автоматически создана из приведенного выше структурированного текстового описания с помощью программы-конвертера. Блок-схема работы генератора диаграмм представлена на рис. 2.

Сначала входное текстовое описание проверяется на отсутствие ошибок. Ошибкой является, например, ссылка на неопределенную метку. В случае обнаружения ошибок пользователю сообщается диагностика и предлагается устранить ошибки. Затем определяется взаимное расположение агентов, а также размеры и координаты агентов, действий, блоков ветвления. В нашем случае агенты упорядочены в порядке первого упоминания во входной спецификации (это являлось требованием заказчика). В общем случае заслуживают внимания другие критерии упорядочивания агентов, например, минимизация суммарной длины соединяющих коннекторов. Затем в диаграмму добавляются коннекторы. Добавление коннектора сводится к определению его начальной и конечной точек. Трассировка между ними осуществляется встроенными средствами MS Visio.

Критерием выбора начальной и конечной точек коннектора является расстояние между ними. Мы стремимся минимизировать это расстояние при сохранении читаемости диаграммы. MS Visio позволяет помещать объекты диаграммы на слои. Мы используем это свойство при работе с аннотациями. Как было указано выше, частью аннотации может быть метка. При наличии метка интерпретируется как идентификатор слоя, на котором должна размещаться аннотация. Аннотации с разными метками помещаются на разные слои. Аннотации без метки считаются комментариями, которые автор спецификации желает видеть на диаграмме. Они помещаются на базовый слой.

Слои являются удобным механизмом детализации базовой спецификации. В случае необходимости детализации для какого-то конкретного приложения название приложения может быть использовано в качестве иденти-

Рис. 2. Схема работы генератора диаграмм

фикатора слоя. Дополнительные данные могут быть определены как аннотации, принадлежащие этому слою. Таким образом, они могут быть отфильтрованы от других имеющихся аннотаций при машинной обработке.

Заключение

Описанный способ преобразования текстового описания в диаграмму последовательности операций реализован и успешно применен на практике для сценариев, описывающих поведение сложных вычислительных систем. Генератор диаграмм использовался не только для визуализации финальной версии сценария, но и как инструмент самоконтроля авторов спецификаций сценариев на промежуточных стадиях. В этом случае диаграмма служила инструментом визуального контроля соответствия написанного текста намерениям автора. Простота структурирования неформализованного текста и использование BPMN в качестве графической нотации способствовали широкому использованию предложенного способа.

Вместе с тем, существует целый ряд направлений для дальнейших улучшений. Одной из задач является анализ качества спецификации в виде структурированного текста, включающий в себя выявление неоднозначности, неполноты, противоречивости. Желательно иметь возможность автоматически определять проблемные места и информировать о них пользователя.

Другой задачей является повышение качества планировки диаграммы последовательности операций. Более компактное расположение элементов диаграммы позволит улучшить восприятие сценариев, состоящих из большого числа действий. Поддержка иерархической спецификации, когда диаграмма сценария верхнего уровня содержит ссылки на подсценарии, позволит создать удобную спецификацию сценариев с повторяющимися подсценариями.

Наконец, путем продуманного добавления аннотаций к базовому описанию сценария (или группы сценариев) можно довести спецификацию до состояния, в котором она может быть использована для автоматизированного создания моделей, пригодных для формального анализа корректности поведения вычислительной системы.

Авторы планируют продолжить исследования по этим направлениям.

Список литературы

1. **Mich L., Franch M., Inverardi P. N.** Market Research for Requirements Analysis Using Linguistic Tools // Requirement Engineering Journal. 2004. N. 9 (1). P. 40—56.

2. **Kiyavitskaya N., Zeni N., Mich L., Berry D. M.** Requirements for Tools for Ambiguity Identification and Measurement in Natural Language Requirements Specifications // Requirement Engineering Journal. 2008. N. 13. P. 207—240.

3. **Hussain I., Ormandjieva O., Kosseim L.** Automatic Quality Assessment of SRS Text by Means of a Decision-Tree-Based Text Classifier // In Proceedings of the 7th International Conference on Quality Software. 2007. P. 209—218.

4. **Tjong S. F., Hallam N., Hartley M.** Improving the Quality of Natural Language Requirements Specifications through Natural Language Requirements Patterns // In Proceedings of the 6th IEEE International Conference on Computer and Information Technology. 2006.

5. **Kamalrudin M., Hosking J., Grundy J.** Improving Requirements Quality Using Essential Use Case Interaction Patterns // In Proceedings of the 33rd International Conference on Sofware Engineering. 2011.

6. **Denger C., Berry D. M., Kamsties E.** Higher Quality Requirements Specifications through Natural Language Patterns // In Proceedings of the IEEE International Conference on Sofware — Science, Technology and Engineering — SwSTE'03. 2003.

7. **Holtmann J., Meyer J., von Detten M.** Automatic Validation and Correction of Formalized, Textual Requirements // In Proceedings of the IEEE 4th International Conference on Sofware Testing, Verification and Validation Workshops — ICSTW. 2011.

8. **Gelhausen T., Tichy W. F.** Thematic Role Based Generation of UML Models from Real Word Requirements // In Proceedings of the 1st IEEE International Conference on Semantic Computing — ICSC. 2007. P. 282—289.

9. **Sharma V. S., Sarkar S., Verma K., Panayappan A., Kass A.** Extracting High-level Functional Design from Software Requirements // In Proceedings of the 16th IEEE Asia-Pacific Software Engineering Conference — APSEC. 2009. P. 35—42.

10. **Deeptimahanti D. K., Sanyal R.** An Innovative Approach for Generating Static UML Models from Natural Language Requirements // Advances in Software Engineering. Communications in Computer and Information Science. 2009. V. 30. P. 147—163.

11. **URL:** http://www.bpmn.org/ Дата обращения: 01.01.2014.

12. **URL:** http://www.omg.org/ Дата обращения: 01.01.2014.

13. **Afreen H., Bajwa I. S.** Generating UML Class Models from SBVR Software Requirements Specifications // In 23rd Benelux Conference on Artifical Intelligence — BNAIC. 2011. P. 23—32.

14. **URL:** http://www.omg.org/spec/SBVR/ Дата обращения: 01.01.2014.

15. **URL:** http://www.omg.org/spec/ Дата обращения: 25.04.2014.

I. S. Pechenko, Research Associate, e-mail: ivan.pechenko@intel.com,
O. V. Venger, Senior Researcher, e-mail: oleg.v.venger@intel.com, ZAO "Intel"

Generation of Flow Diagram from Textual Description of the Flow

Nowadays computer systems design is a very complex process, and the specification of components interaction is a serious part of the design process. The quality of the specification is the important factor for system design and test efficiency. Commonly used model of system specification is a set of scenarios (or flows) of components interaction. There is no common standard for flow description, and now 95 % of such specifications are textual scenarios descriptions. They are commonly inaccurate, inconsistent, incomplete and cannot be automatically processed. Here in this work we propose a method of converting textual flow specifications to a form with better readability, usable for automatic processing and more convenient for data sharing between project stakeholders. Special tags are added to the textual flow description to structure it, and then flow diagram is automatically generated from the text using BPMN notation. This method was implemented and successfully used for improving complex systems specifications.

Keywords: *behavior model, behavior scenario, structured text, automatic specification processing, flow diagram, BPMN*

References

1. **Mich L., Franch M., Inverardi P. N.** Market Research for Requirements Analysis Using Linguistic Tools. *Requirement Engineering Journal*. 2004. N. 9 (1). P. 40—56.

2. **Kiyavitskaya N., Zeni N., Mich L., Berry D. M.** Requirements for Tools for Ambiguity Identification and Measurement in Natural Language Requirements Specifications. *Requirement Engineering Journal*. 2008. N. 13. P. 207—240.

3. **Hussain I., Ormandjieva O., Kosseim L.** Automatic Quality Assessment of SRS Text by Means of a Decision-Tree-Based Text Classifier. *In Proceedings of the 7th International Conference on Quality Software*. 2007. P. 209—218.

4. **Tjong S. F., Hallam N., Hartley M.** Improving the Quality of Natural Language Requirements Specifications through Natural Language Requirements Patterns. *In Proceedings of the 6th IEEE International Conference on Computer and Information Technology*. 2006.

5. **Kamalrudin M., Hosking J., Grundy J.** Improving Requirements Quality Using Essential Use Case Interaction Patterns. *In Proceedings of the 33rd International Conference on Sofware Engineering*. 2011.

6. **Denger C., Berry D. M., Kamsties E.** Higher Quality Requirements Specifications through Natural Language Patterns. In *Proceedings of the IEEE International Conference on Sofware — Science, Technology and Engineering — SwSTE'03*. 2003.

7. **Holtmann J., Meyer J., von Detten M.** Automatic Validation and Correction of Formalized, Textual Requirements. *In Proceedings of the IEEE 4th International Conference on Sofware Testing, Verification and Validation Workshops — ICSTW*. 2011.

8. **Gelhausen T., Tichy W. F.** Thematic Role Based Generation of UML Models from Real Word Requirements. *In Proceedings of the 1st IEEE International Conference on Semantic Computing — ICSC*. 2007. P. 282—289.

9. **Sharma V. S., Sarkar S., Verma K., Panayappan A., Kass A.** Extracting High-level Functional Design from Software Requirements. *In Proceedings of the 16th IEEE Asia-Pacific Software Engineering Conference — APSEC*. 2009. P. 35—42.

10. **Deeptimahanti D. K., Sanyal R.** An Innovative Approach for Generating Static UML Models from Natural Language Requirements. *Advances in Software Engineering. Communications in Computer and Information Science*. 2009. V. 30. P. 147—163.

11. **URL:** http://www.bpmn.org/ Дата обращения: 01.01.2014.

12. **URL:** http://www.omg.org/ Дата обращения: 01.01.2014.

13. **Afreen H., Bajwa I. S.** Generating UML Class Models from SBVR Software Requirements Specifications. In 23rd Benelux Conference on Artifical Intelligence — BNAIC. 2011. P. 23—32.

14. **URL:** http://www.omg.org/spec/SBVR/ Data obrasheniya: 01.01.2014.

15. **URL:** http://www.omg.org/spec/ Data obrasheniya: 25.04.2014.

УДК 004.912

М. В. Бочков, д-р техн. наук, проф.,
НОУ ДПО "Центр предпринимательских рисков", г. Санкт-Петербург,
П. Н. Бойков, вед. специалист,
ОАО "НИИ "Рубин", г. Санкт-Петербург, e-mail: boykova@yandex.ru

Инфодинамическая модель поиска пользователя в социальной сети

Исследованы закономерности представления пользователями соцсетей своих данных, на основе которых разработан алгоритм формирования оптимальной стратегии поиска в социальных сетях.

Ключевые слова: *социальная сеть, поиск в социальной сети, алгоритм оптимизации поиска в социальных сетях, инфодинамическая модель*