Национальный исследовательский
Томский государственный университет
Кемеровский государственный университет
Кемеровский научный центр СО РАН
Институт вычислительных технологий СО РАН
Филиал Кемеровского государственного университета
в г. Анжеро-Судженске

ИНФОРМАЦИОННЫЕ ТЕХНОЛОГИИ И МАТЕМАТИЧЕСКОЕ МОДЕЛИРОВАНИЕ (ИТММ–2014)

Материалы XIII Международной научно-практической конференции имени А. Ф. Терпугова

20–22 ноября 2014 г.

Часть 1

Издательство Томского университета

2014

мирования, поэтому выбран универсальный источник данных – диаграммы UML. Стоит отметить, что работа не должна ограничиваться диаграммой классов, для полных и достоверных результатов при анализе необходимо использовать диаграммы последовательности и др.

Использование диаграмм UML позволит производить анализ уже на этапе проектирования будущего приложения. Таким образом, многие серьезные ошибки проектирования будут исправлены еще до начала этапа разработки.

Выбор подхода для поиска артефактов является целью дальнейших исследований.

Литература

1. *Destefanis G.* Assessing Software Quality by Micro Patterns Detection [Electronic resource]. – URL: http://veprints.unica.it/859/3764 (access date: 14.05.2014).

2. *Heuzeroth D.* Automatic Design Pattern Detection / D. Heuzeroth, T. Holl, G. Högström, W. Löwe [Electronic resource]. – URL: http://www.eecs.yorku.ca/ course_ archive/ 2005-06/W/6431/Heuzeroth.pdf (access date: 01.05.2014).

3. *Rao R. S.* Design Pattern Detection by Multilayer Neural Genetic Algorithm / R. S. Rao, M. Gupta [Electronic resource]. – URL: http://ijcsn.org/articles/0301/Design-Pattern-Detection-by-Multilayer-Neural-Genetic-Algorithm.html (access date: 05.05.2014).

4. *Bergenti F.* Improving UML Designs Using Automatic Design Pattern Detection. F. Bergenti, A. Poggi [Electronic resource]. – URL: http://citeseerx.ist.psu.edu/viewdoc/ summary?doi=10.1.1.22.3764 (access date: 01.06.2014).

5. *Meyer M.* Pattern-based reengineering of software systems // WCRE ’06: Proceedings of the 13th Working Conference on Reverse Engineering. – Washington, DC, USA, 2006. – P. 305–306.

ЭВОЛЮЦИОННЫЙ ПРОТОТИП ИНСТРУМЕНТА ДЛЯ РАБОТЫ С ВАРИАНТАМИ ИСПОЛЬЗОВАНИЯ

О. А. Змеев, Я. М. Чайка

Национальный исследовательский

Томский государственный университет, Томск, Россия

Процедура определения требований к программной системе является довольно важным этапом в любом процессе разработки. Для фиксации функциональных требований к программному обеспечению могут использоваться варианты использования [1]. При этом зачастую возникает необходимость ознакомить заказчика (или другое заинтересованное лицо) с функциональными требованиями, оформленными в виде вариантов использования. Однако заинтересованное лицо не всегда обладает достаточными знаниями нотаций, используемых для записи вариантов использования. Потому единственным способом представления функциональных требований для заказчика является текстовое описание на естественном языке, для понимания которого не требуется знаний специализированных нотаций.

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

Разнообразие способов представления сценариев порождает проблему отсутствия механизма автоматического преобразования между различными способами записи. Отсутствие такового приводит к рутинной работе по спецификации одного и того же варианта использования несколько раз, что может стать причиной возникновения ошибок (из-за расхождений между представлениями).

В качестве решения данной проблемы был предложен следующий подход: создание унифицированной модели вариантов использования, поддерживающей необходимые форматы представления вариантов использования в достаточном объеме, позволяющем их специфицировать [2].

Избавившись при использовании предложенной модели от трудоемкой реализации преобразований «из каждого в каждое», можно задавать собственные форматы представления сценария варианта использования. При этом необходимо будет обладать знаниями только нотации унифицированной модели и собственного способа спецификации, и нужно будет осуществить всего два преобразования – из унифицированной модели в свою нотацию и наоборот.

Были рассмотрены следующие существующие средства, позволяющие моделировать функциональные требования к системам и специфицирования их в виде вариантов использования:

•Microsoft Visual Studio [3]. Имеется возможность работы с полной моделью вариантов использования в виде схемы вариантов использования. Специфицирование вариантов использования происходит при помощи связывания варианта использования с другой схемой или схемами проекта, например схемой деятельности или схемой последовательности, созданными в том же программном продукте, или же связать со сторонним файлом, например страницей OneNote или документом MS Word. Преобразование между текстовыми и графическими форматами не предусматривается.

•Visual Paradigm [4]. Имеется возможность работы с полной моделью вариантов использования. Спецификация варианта использования происходит в текстовом формате, схожем на формат сценария варианта использования, предложенного A. Cockburn, и есть синхронизация текстового сценария с диаграммами деятельности, последовательности, коммуникаций. Обратного автоматического преобразования не предусмотрено.

•Rational Rose Enterprise [5]. Аналогично Microsoft Visual Studio имеется поддержка работы с полной моделью вариантов использования, и их спецификация также подразумевает прямое связывание с файлом. Механизма синхронизации не было обнаружено.

Вышеперечисленные средства моделирования функциональных требований не поддерживают механизмов автоматического двустороннего преобразования между представлениями и предоставляют довольно ограниченный набор форматов специфицирования вариантов использования, который не во всех средствах можно расширить посредством добавления дополнительного модуля.

Целью данной работы является разработка эволюционного прототипа инструмента для работы с вариантами использования, который бы основывался на унифицированной модели, поддерживал возможность работы с полной моделью вариантов использования и имел точку расширения для добавления произвольного способа спецификации варианта использования, описанного в качестве модуля расширения (плагина) к системе.

Таким образом, пользователи смогли бы без особых затрат по времени настраивать произвольное, удобное для себя представление сценария варианта использования, но при этом имели возможность получения версий своего сценария во всех остальных подключенных в виде плагинов способов спецификации варианта использования, а также получали возможность отображать связи между вариантами использования на диаграммах вариантов использования и получать полное описание документируемой таким способом разрабатываемой системы.

Для отображения понятий предметной области использовались следующие определения:

•Плагин – модуль расширения, добавляющий новый способ представления сценария варианта использования. Плагин не требует модификации расширяемой системы и подключается динамически.

•Проект – объединение диаграмм и сценариев вариантов использования.

•Диаграмма – совокупность вариантов использования, актеров и связей между ними – расширения, включения, обобщения и ассоциации.

С каждым из определений связан свой набор требований:

•Управление проектами (операции добавления, удаления, переименования проекта и т.д.).

•Управление диаграммами (операции создания, редактирования названия и элементов диаграммы и т.д.).

•Управление плагинами (операции добавления и исключения плагина из системы и т.д.).

•Управление вариантами использования (операции добавления варианта использования, управление его сценарием, в том числе добавление расширений и т.д.).

Для программной реализации перечисленных требований было спроектировано архитектурное решение (рис. 1).

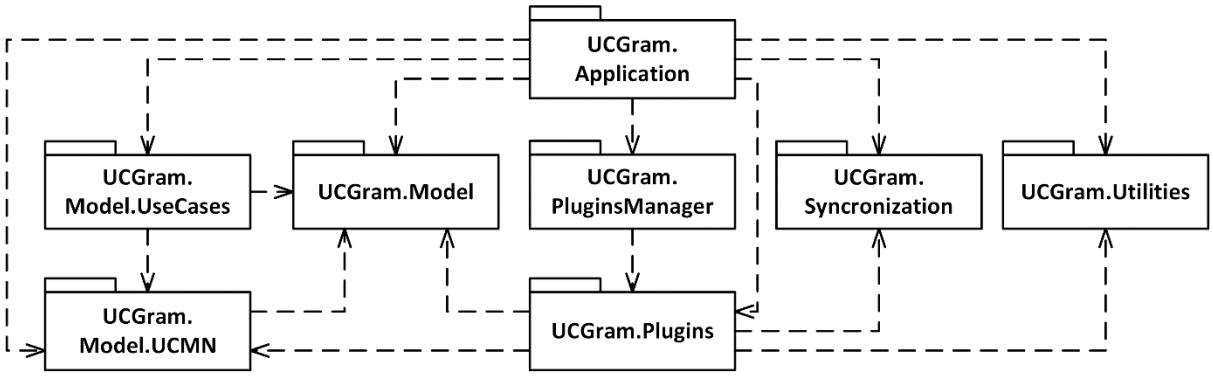

Рис. 1. Диаграмма пакетов ядра редактора вариантов использования

Основные пакеты редактора:

•UCGram.Application: отвечает за отображение всего редактора в целом. Является частью, реализующей представление и модель представления в шаблоне MVVM.

•UCGram. Model: содержит базовый класс для предоставления интерфейса уведомления и уникальной идентификации объектов внутри приложения.

•UCGram.Model.UseCases: отражает предметную область, связанную с элементами диаграммы вариантов использования.

•UCGram.Model.UCMN: отражает предметную область относительно графической нотации унифицированной модели вариантов использования.

•UCGram.Plugins: содержит базовые классы для разработки плагинов, содержащих дополнительные способы представления спецификации сценария варианта использования.

•UCGram.PluginsManager: содержит классы, отвечающие за получение информации о доступных плагинах и подключение этих плагинов в систему.

•UCGram.Utilities – содержит вспомогательные классы для реализации команд, базовые классы для обработки действий над представлениями диаграмм вариантов использования.

•UCGram.Synchronization – содержит базовые классы для синхронизации представлений сценариев вариантов использования. Должен использоваться разработчиком для реализации механизма синхронизации разрабатываемых им представлений и представлений в нотации унифицированной модели.

В рамках системы предусмотрено нераздельное сохранение проекта вместе с входящими в него диаграммами и сценариями вариантов использования посредством технологии LINQ to XML, которая позволяет эффективно работать с файлами формата XML, создавая, загружая, обрабатывая и сохраняя документ как XML-дерево.

Для разрабатываемого редактора также был разработан модуль расширения, поддерживающий текстовую двухколоночную нотацию Р. Вирфс-Брок [6], для реализации которого необходимо было импортировать следующие пакеты основного приложения: UCGram.Plugins, UCGram.Utilities, UCGram.Synchronization, UCGram.Model, UCGram. Model.UCMN.

Для плагина в текстовой нотации спроектирована архитектура (рис. 2).

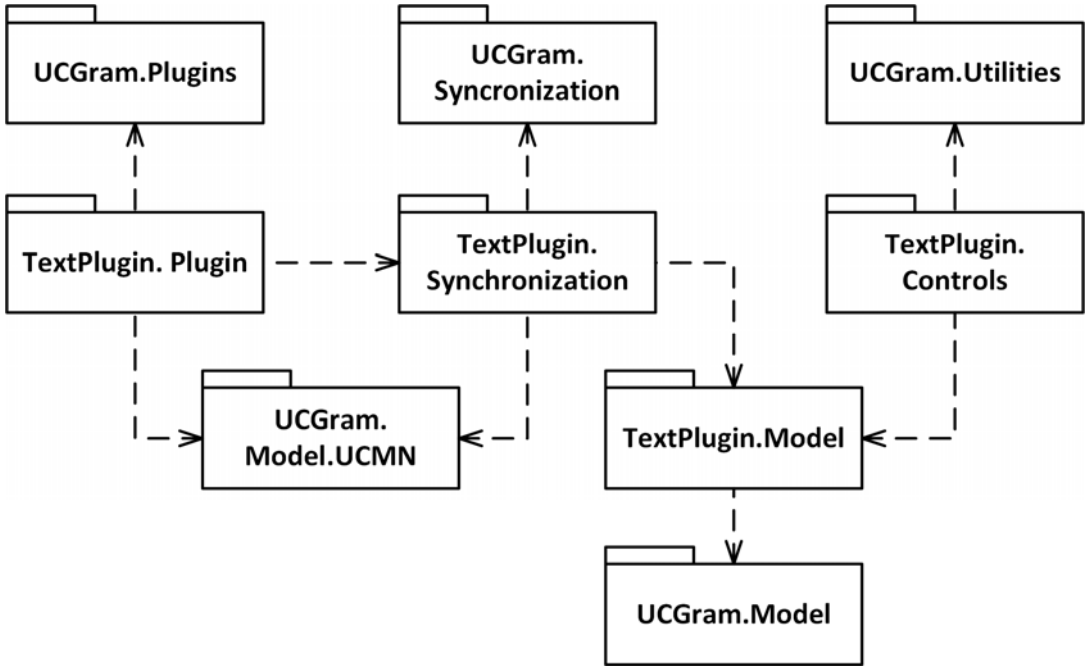

Рис. 2. Диаграмма пакетов для плагина TextPlugin

Пакет TextPlugin. Plugin отвечает за подключение плагина к основному приложению; TextPlugin. Synchronization определяет двух менеджеров синхронизации, необходимых для двустороннего преобразования изменений; TextPlugin. Controls отвечает за отображение табличного представления и осуществления действий над ним; TextPlugin. Model отражает предметную область относительно текстовых нотаций Вирфс-Брок.

В результате текущей работы был реализован и протестирован эволюционный прототип инструмента для работы с вариантами использования, удовлетворяющий функциональным требованиям, принятым базовыми на данном этапе разработки, а также реализован текстовый плагин для расширения набора представлений. В дальнейшем планируется работа в направлении преобразования вариантов использования в модель анализа и модель тестирования.

Литература

1. *Jacobson I.* Modeling with Use Cases: Formalizing use-case modeling // JOOP. – June 1995. – Vol. 8, No. 3.

2. *Политов А. М., Чайка Я. М.* Унифицированная модель варианта использования // Материалы 51-й Международной научной студенческой конференции «Студент и научно-технический прогресс»: Информационные технологии. – Новосибирск: Новосиб. гос. ун-т, 2013. – С. 209.

3. Microsoft Developer Network [Электронный ресурс]. – URL: http:// msdn. microsoft.com/ en-us/library/dd409432.aspx (дата обращения: 30.05.2014).

4. Software Design Tools for Agile Teams, with UML, BPMN and More: [Электронный ресурс]. – URL: http://www.visual-paradigm.com (дата обращения: 23.03.2014).

5. IBM – Rational Rose Enterprise [Электронный ресурс]. – URL: http://www-03.ibm.com/software/products/ru/enterprise (дата обращения: 30.03.2014).

6. *Wirfs-Brock R.* Designing Scenarios: Making the Case for a Use Case Framework // The Smalltalk Report. – November-December 1993. – Vol. 3, No. 3.

КОМПЛЕКС ПРОГРАММНОГО ОБЕСПЕЧЕНИЯ, ИСПОЛЬЗУЕМОГО В ПРОЦЕССЕ ТЕСТИРОВАНИЯ И ЭКСПЛУАТАЦИИ ВЫЧИСЛИТЕЛЬНЫХ ИЗДЕЛИЙ АВИОНИКИ НА ОСНОВЕ ИНТЕРФЕЙСА RS-232

П. В. Коновалов, С. Б. Уткин, С. В. Батова

Санкт-Петербургское опытно-конструкторское бюро «Электроавтоматика», Санкт-Петербург, Россия

Введение

Постоянное развитие электроники и появление новых вычислительных компонентов даёт возможность создавать всё более совершенные вычислительные устройства, используемые в составе бортовых вычислительных систем. В процессе разработки новых аппаратных средств возникает необходимость проводить тестирование как всей системы, так и отдельных её компонентов [1], при этом желательно получение результатов в форме наглядных сообщений и/или диаграмм. Имея возможность проведения автоматизированной отладки с получением подробных и удобочитаемых результатов, можно существенно повысить качество итогового изделия и сократить количество времени, затраченного на ввод устройства в эксплуатацию.

Использование ПК при создании средств авионики позволяет облегчить процессы разработки, отладки и последующего обслуживания оборудования. Однако подключение бортовых вычислителей к стационарному компьютеру требует использования специализированных интерфейсов и протоколов, обеспечивающих взаимодействие аппаратных платформ. Также необходима некоторая программная оболочка, позволяющая пользова-