МИНИСТЕРСТВО ОБРАЗОВАНИЯ И НАУКИ РОССИЙСКОЙ ФЕДЕРАЦИИ
Томский государственный университет
Горно-Алтайский государственный университет
Институт оптики атмосферы им. В.Е. Зуева СО РАН

НОВЫЕ ИНФОРМАЦИОННЫЕ ТЕХНОЛОГИИ В ИССЛЕДОВАНИИ СЛОЖНЫХ СТРУКТУР

МАТЕРИАЛЫ ДЕСЯТОЙ РОССИЙСКОЙ КОНФЕРЕНЦИИ С МЕЖДУНАРОДНЫМ УЧАСТИЕМ

Томск
Издательский Дом Томского государственного университета
2014

ностям-классам соответствуют такие динамические объекты жизненного цикла, как роль, задача, артефакт, руководство, шаблон задачи.

Метамодель SPEM имеет две реализации: EPF Composer и IBM Rational Method Composer. Первый программный продукт является бесплатным и позволяет создавать описания процесса разработки ПО в соответствии с метамоделью. Второй программный продукт является платным, но имеет более широкий функционал. Главное отличие данного приложения – возможность генерировать на основе описаний объекты жизненного цикла в инструменте управления проектами IBM Rational Team Concert.

Согласно программной документации [4], в IBM Rational Method Composer (rmc) можно осуществлять генерацию следующих динамических объектов: роль на основе rmc-роли, задача на основе rmc-задачи, шаблон задачи на основе rmc-описания процесса, тип задачи на основе специально-определенного rmc-описания процесса. Все отношения между объектами, артефакты, категоризации, ограничения целостности и рабочие потоки необходимо создавать вручную.

Таким образом, ни одна из реализаций метамодели SPEM не поддерживает полную генерацию объектов жизненного цикла. Реализация полной генерации объектов на основе описаний является целью дальнейших исследований.

Литература

1. *Geisler R., Klar M., Pons C.* Dimensions and Dichotomy in Metamodeling. URL: http://citeseerx.ist.psu.edu/ viewdoc/summary?doi=10.1.1.200.3534 (access date: 01.12.2013).
2. *Буч Г., Рамбо Д., Якобсон И.* Язык UML. Руководство пользователя. 2-е изд. / пер. с англ. Н. Мухин. М. : ДМК Пресс. 496 с.
3. *Software* & Systems Process Engineering Meta-Model Specification. URL: http://www.omg.org/spec/SPEM/2.0.html (access date: 22.09.2013).
4. *IBM* Rational Method Composer. Product documentation. URL: http://www-01.ibm.com/support/ docview.wss?uid=swg27027354 (access date: 06.01.2014).

ИСПОЛЬЗОВАНИЕ УНИФИЦИРОВАННОЙ МОДЕЛИ ВАРИАНТОВ ИСПОЛЬЗОВАНИЯ ДЛЯ ФИКСАЦИИ ФУНКЦИОНАЛЬНЫХ ТРЕБОВАНИЙ К ПРОГРАММНЫМ СИСТЕМАМ

О.А. Змеев, А.М. Политов, Я.М. Чайка

Национальный исследовательский Томский государственный университет, Томск, Россия
ozmeev@gmail.com, a.m.politov@gmail.com, chaykayana@gmail.com

В процессах разработки программного обеспечения, в основе которых лежит объектно-ориентированная парадигма, для определения функциональных требований и их последующей фиксации применяются варианты использования [1]. Существует довольно много способов спецификации вариантов использования [2]. Заказчики предпочитают использовать текстовые описания на естественном языке, поскольку для использования диаграмм необходимы дополнительные знания нотаций. Разработчики чаще используют графические представления в виде диаграмм, поскольку такие представления имеют формальную модель, которая может использоваться для дальнейших преобразований в другие необходимые артефакты процесса разработки.

Достоинства текстового и графического способов описания варианта использования приводят к необходимости вручную создавать требуемые представления, и, впоследствии, выполнять синхронизацию изменений. Для больших программных систем соответствующее преобразование, с одной стороны, и поддержка различных представлений вариантов использования в актуальном состоянии, с другой, ведет к достаточно серьезным издержкам.

Для решения данной проблемы предлагается разработать унифицированную модель вариантов использования, которая должна поддерживать различные способы представления.

В основе унифицированной модели лежит следующая концепция – спецификация варианта использования происходит при помощи разработанной графической нотации модели, основанной на диаграммах деятельности UML. При необходимости получения описания требования в другой нотации запускается механизм преобразования спецификации варианта использования из формата унифицирован-

ной модели в формат нужного представления. На текущий момент модель поддерживает двухколоночный табличный формат текстовых спецификаций, разработанный R. J. Wirfs-Brock [3]. Предлагаемая модель представляет собой расширение к модели UML (версии 2.4.1, стандартизованной ISO) [4].

Для актуализации зафиксированного функционального требования необходимо внести изменения только в его представление в нотации унифицированной модели, из которого можно получить актуальные версии других форматов представления варианта использования.

Для предложенной модели был разработан эволюционный прототип инструмента, позволяющего формально зафиксировать функциональные требования к системе, преобразовать их в необходимое представление и в дальнейшем использовать их в процессе разработки.

В дальнейших планах расширение унифицированной модели до поддержки полной модели вариантов использования и увеличение набора поддерживаемых представлений варианта использования. Также планируется исследовать возможность трассировки вариантов использования в формате унифицированной модели в последующие артефакты процесса разработки – модель анализа и модель тестирования.

Литература

1. *Jacobson I.* Modeling with Use Cases: Formalizing use-case modeling // JOOP. 1995. June. Vol. 8, No. 3.
2. *Hurlbut R.R.* A Survey of Approaches For Describing and Formalizing Use Cases. Electronic data. Expertech, Wheaton, Illinois, 1997. URL: http://www.iit.edu/~rhurlbut/xpt-tr-97-03.html (access: 14.12.2013).
3. *Wirfs-Brock R.* Designing Scenarios: Making the Case for a Use Case Framework. // The Smalltalk Report. 1993. November-December. Vol. 3, No. 3.
4. *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.

ВЕРОЯТНОСТНО-ВРЕМЕННЫЕ ХАРАКТЕРИСТИКИ ТРЁХУРОВНЕВОЙ ПАМЯТИ НЕБЛОКИРУЮЩЕГО ТИПА

Н.А. Иванов, С.П. Сущенко

Национальный исследовательский Томский государственный университет, Томск, Россия
nicktorwald@gmail.com, ssp.inf.tsu@gmail.com

В современной организации подсистемы памяти, где каждый уровень может работать независимо и асинхронно, возможна работа памяти в виде конвейера, где на каждом уровне может обрабатываться отдельная транзакция доступа к памяти. Это позволяет, в случае неуспешного доступа к участку памяти в рамках кэша первого уровня, не дожидаться окончательной обработки текущей транзакции на других уровнях памяти и начать обработку следующей. Количество одновременно выполняемых транзакций к памяти физически ограничивается межуровневым буфером, хранящим результаты транзакций. Кэш-память с подобным поведением называется *кэшем неблокирующего типа*, а объём межуровневого буфера – *глубиной неблокируемости кэша* N.

Работа трёхуровневой памяти описывается с помощью двумерной цепи Маркова с дискретным временем и длительностью шага, равной t. Первая компонента I описывает количество этапов обработки для кэш-памяти второго уровня, вторая компонента J – количество этапов обработки для ОЗУ, тогда P_{ij} – вероятность нахождения системы в состоянии $(i,j), i \in I, j \in J$. Будем полагать, что ЦП порождается неограниченный поток обращений к подсистеме памяти и вероятности промаха для кэша первого и второго уровней составляют R_1 и R_2 соответственно, тогда интенсивности входных потоков выражаются, как $\Lambda_1 = K_1 R_1 \Lambda_0$ и $\Lambda_2 = (K_2 R_2 / K_1)\Lambda_1$ для кэша второго уровня и ОЗУ соответственно. Где K_1 и K_2 количество порождаемых этапов обработки кратных t и $1 < K_1 < K_2$ на соответствующих этапах.

Определим несколько важнейших характеристик для трёхуровневой подсистемы памяти. Первая из них *вероятность блокировки*, определяющая долю времени, в течении которого не доступен межуровневый интерфейс (новые запросы к памяти от ЦП не поступают):

$$Q(N, K_1, K_2) = \sum_{(i,j) \in S'} P_{ij}\,,\ S' = \left\{ (i,j) \middle| i,j \in \mathbb{N} \wedge \left\lceil {}^{i}/_{K_1} \right\rceil + \left\lceil {}^{j}/_{K_2} \right\rceil = N \right\}.$$