УДК 004.94

Моделирование графического редактора системы *Insertion Modeling System* средствами *Graphical Modeling Framework Eclipse*

В.С. Песчаненко

Приведены основные сведения о системе инсерционного моделирования *Insertion Modeling System* и о технологии *Graphical Modeling Framework Eclipse*. Представлен процесс моделирования графических редакторов средствами *Eclipse*. Описаны диаграмма классов для графического редактора *IMS* и пример модели транзиционной системы в этом редакторе.

The main notions about the Insertion Modeling System and the GMF framework Eclipse are described. The process of creation of the graphical editor in GMF Eclipse is presented. The diagram of the classes for the IMS graphical editor is described. The model of the transition system in this editor is presented.

Наведено основні відомості про систему інсерційного моделювання *Insertion Modeling System*, технологію *Graphical Modeling Framework Eclipse*. Представлено процес моделювання графічних редакторів засобами *Eclipse*. Описано діаграму класів для графічного редактора *IMS* та приклад транзиційної системи у цьому редакторі.

Введение. Система инсерционного моделирования (*Insertion Modelling System – IMS*) была разработана на базе системы алгебраического программирования *APS* – первой системы переписывания термов, которая отделила понятия стратегии от систем переписывающих правил [1, 2]. Прототип системы инсерционного моделирования был разработан на языке *APLAN* в 1999 году в отделе 100 Института кибернетики имени В.М. Глушкова НАН Украины [3] под руководством академика А.А. Летичевского. Его использование было затруднено ошибками работы с памятью в *APS*, исправленными в 2004 году в Научно-исследовательском институте информационных технологий Херсонского государственного университета [4]. В 2010 году прототип системы был перенесен с языка верхнего уровня *APLAN* (язык системы *APS*) на язык нижнего уровня *C++* в *APS*, после чего авторами было принято решение о создании новой системы – *IMS*, на основе *APS*.

Однако использование системы *IMS* на данном этапе ее развития затруднено, поскольку:

- для написания собственных программ в *IMS* необходимо знания языка *APLAN* системы *APS*;
- использование алгебраических структур данных [5] приводит к потере наглядности;
- модель достаточно сложно понять человеку, который ее не писал.

Поэтому наглядность и простота в использовании системы *IMS* актуальна для ее пользователей. Для того чтобы решить эти проблемы, создан графический редактор транзиционных систем. Конечно, в полной мере созданный редактор не может решить проблемы знания языка *APLAN*, однако он сможет свести их к необходимому языковому минимуму, а также даст необходимый инструментарий для создания соответствующих моделей.

В качестве технологии для реализации графического редактора мы выбрали технологию *GMF* (*Graphical Modelling Framework*) [6] многокомпонентной среды *Eclipse* [7].

Таким образом, данная статья посвящена описанию процесса моделирования графического редактора средствами *GMF Eclipse*, а также его применению для создания моделей в системе *IMS*.

Инсерционное моделирование

Инсерционное моделирование – направление, развивающееся на протяжении последнего десятилетия как подход к исследованию распределенных многоагентных систем и разработке средств верификации распределенных параллельных программ и аппаратуры [8].

Общепринятым средством для описания динамики систем в современной компьютерной науке есть понятие транзиционной системы, т.е. системы, определяемой множеством состояний

и отношением переходов. Обычно это понятие обогащается добавлением различных дополнительных структур, важнейшая из которых – разметка переходов (размеченные транзиционные системы, введенные Парком [9] для описания поведения автоматов на бесконечных словах). В инсерционном моделировании исходным понятием есть понятие атрибутной транзиционной системы [8], которое формально определяется как пятерка $< S, A, U, T, \varphi >$, где S – множество состояний, A – множество действий, используемых для разметки переходов, U – множество атрибутных разметок, используемых для разметки состояний, T – отношение переходов, $T \subseteq S \times A \times S \cup S \times S$, состоит из размеченных переходов $s \xrightarrow{a} s'$ и неразмеченных переходов $s \to s'$. Эта часть структуры соответствует обычному понятию размеченной транзиционной системы (со скрытыми переходами).

Функция $\varphi : S \to U$ называется функцией разметки состояний. Обычно U определяется как множество $U = D^R$ отображений множества R атрибутов в множество данных D (область значений атрибутов) или в случае типизированных данных как семейство $U = (D_\xi^{R_\xi})_{\xi \in \Xi}$, где Ξ есть множество типов данных. Обычно это интерпретированный или неинтерпретированный язык первого порядка, возможно расширенный некоторыми модальностями темпоральной логики.

Транзиционные системы могут также настраиваться путем выделения некоторых специфических множеств состояний из множества состояний S. Среди них важнейшие – множества S_0 начальных, S_Δ заключительных и $S_\perp$ неопределенных состояний. Последние используются в теории для определения отношения апроксимации и построения бесконечных систем в виде пределов конечных.

В инсерционном моделировании в качестве инвариантов используются (в общем случае бесконечные) выражения или системы уравнений в алгебре поведений. Алгебра поведений устроена достаточно просто. Она представляет собой двухосновную алгебру $< U, A >$, первой компонентой которой есть множество U поведений, а вторая представляет множество действий A. Сигнатура алгебры поведений состоит из двух операций, одного отношения и трех констант. Первая операция $a.u$ называется префиксинг. Ее аргументами являются действие a и поведение u. Результат – новое поведение. Вторая операция – операция недетерминированного выбора $u + v$. Это бинарная операция, определенная на множестве поведений. Она коммутативна, ассоциативна и идемпотентна. Константы алгебры поведений – успешное завершение Δ, неопределенное поведение $\perp$ и тупиковое поведение 0, которое является нулем (нейтральным элементом) недетерминированного выбора. На множестве поведений определено бинарное отношение аппроксимации $\subseteq$, т.е. отношение частичного порядка с наименьшим элементом $\perp$. Операции префиксинга и недетерминированного выбора монотонны и непрерывны относительно этого отношения. Существенную роль играет полная алгебра поведений $F(A)$, содержащая пределы всех направленных множеств и, следовательно, в ней имеет место теорема о наименьшей неподвижной точке. Точная конструкция алгебры $F(A)$ (для произвольного, в том числе и бесконечного множества действий) изложена в [10].

В полной алгебре поведений каждый элемент имеет представление

$$u = \sum_{i \in I} a_i.u_i + \varepsilon_u ,$$

которое определяется единственным образом (с точностью до коммутативности и ассоциативности), если все $a_i.u_i$ различны.

С каждым состоянием s транзиционной системы S связывается поведение $\mathrm{beh}(s) = u_s$ системы S в этом состоянии как компонента наименьшего решения системы уравнений

$$u_s = \sum_{s \xrightarrow{a} t} a.u_t + \varepsilon_s ,$$

где $\varepsilon_s = 0, \Delta, \perp, \Delta + \perp$ в зависимости от выполнения условий $s \notin S_\Delta \cup S_\perp$, $s \in S_\Delta \setminus S_\perp$, $s \in S_\perp \setminus S_\Delta$, $s \in S_\Delta \cup S_\perp$, соответственно.

Алгебра поведений обогащается двумя композициями [10]:

• последовательной

$$uv = \sum_{u \xrightarrow{a} u'} a.(u'v) + \sum_{u=u+\varepsilon} \varepsilon v,$$

$$0v = 0,\ \Delta v = \Delta,\ \bot v = \bot,\ a \mapsto a.\Delta = (a;\Delta) = a\Delta;$$

• параллельной

$$u \,\|\, v = \sum_{\substack{u \xrightarrow{a} u' \\ v \xrightarrow{b} v'}} (a \times b).(u' \,\|\, v') + \sum_{u \xrightarrow{a} u'} a.(u' \,\|\, v) +$$

$$+ \sum_{v \xrightarrow{b} v'} b.(u \,\|\, v') + (\varepsilon_u \,\|\, \varepsilon_v),$$

$$\varepsilon \,\|\, \varepsilon' = \varepsilon' \,\|\, \varepsilon, \varepsilon \,\|\, \Delta = \varepsilon, \varepsilon \,\|\, \bot = \bot, 0 \,\|\, \Delta = 0 \,\|\, 0 = 0.$$

GMF Eclipse

Среда *Eclipse* – это кросс-платформенная интегрированная среда разработки программного обеспечения с открытыми исходными кодами. Главный язык разработки, поддерживаемый этой средой, – *Java*, хотя имеется также поддержка *C++*, *Perl*, *Fortran* и др. На базе этой среды создаются различные дополнительные технологии, одна из которых – *Eclipse Graphical Modeling Framework* (*GMF*) [6], первая ее версия появилась в середине 2006 года.

Технология *GMF* предназначена для быстрой разработки графических средств, главным образом интегрируемых в *Eclipse*. Она является *Open Source* разработкой и развивается, в основном, специалистами компаний *IBM* и *Borland*. *GMF* интегрирует две широко используемые и известные *Eclipse*-библиотеки – *Eclipse Modeling Framework* (*EMF*) и *Graphical Editing Framework* (*GEF*). Архитектура *DSM*-пакета с применением *GMF* строится на основе *MVC*-шаблона. Для создания уровней представления и контроллеров используется технология *GEF*, для создания моделей – технология *EMF*.

Процесс разработки *DSM*-пакетов на основе *GMF* изображен на рис. 1 и состоит из следующих шагов [11].

• Разработка доменной модели (*domain model*) – модели целевой предметной области, для которой предназначается создаваемый графический редактор. Эта модель – метамодель нового *DSL*. Доменная модель разрабатывается с помощью графического редактора *GMF Ecore*. Выходной файл с описанием модели – **.ecore*.

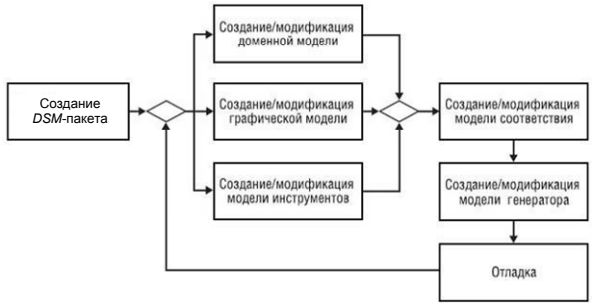

Рис. 1. Схема работы с технологией *GMF Eclipse*

• Разработка графической модели (*graphical definition*) – описания графической нотации создаваемого языка. Эта модель может быть сгенерирована автоматически и изменена вручную. Выходной файл с описанием этой модели – **.gmfgraph*.

• Разработка модели инструментов (*tool definition*) – описания элементов панели инструментов будущего редактора (палитры графических объектов, списка действий, меню графических объектов и пр.). Эта модель может быть сгенерирована автоматически и изменена вручную. Выходной файл с описанием этой модели – **.gmftool*.

• Разработка модели соответствия (*mapping model*). Все предыдущие модели являются независимыми, формально никак не связанными друг с другом. Каждая из них располагается в одном или нескольких отдельных файлах. В них определяется то, что может использоваться при построении диаграммного редактора, а что будет использоваться и как именно – определяется в модели соответствия. Для элементов доменной модели определяются связи с графическим представлением, а также соответствующими инструментами. Не все элементы из доменной модели могут попасть в модель соответствия, некоторые же могут использоваться по нескольку раз. Выходной файл этой модели – **.gmfmap*.

• Создание модели генератора (*generator model*) – описания, по которым производится генерация целевого графического редактора. Данная модель – промежуточное представление будущего редактора. Она автоматически генерируется по модели соответствия и дополняется разработчиками «вручную». Примеры информации, содержащейся в ней: свойства редактора (его

имя, расширение диаграммных файлов), настройки генератора редакторов *GMF*. Выходной файл модели – **.gmfgen*.

• Генерация кода целевого *DSM*-пакета.

Графический редактор *IMS*

Требования пользователя к графическому редактору *IMS*:

должен поддерживать графическое представление:

1) состояния транзиционной системы *S*, в том числе начального состояния, произвольного состояния, поведений Δ, $\perp$, 0;

2) операции префиксинга «.» и недетерминированного выбора «+»;

3) транзиционной системы, последовательной и параллельной композиций с учетом вложенности этих понятий (последовательная и параллельная композиции тоже являются транзиционной системой);

4) состояния среды и функции погружения с учетом вложенности (погруженные в среду агенты тоже являются состоянием среды).

Рассмотрим доменную модель созданного графического редактора *IMS* (рис. 2) и покажем, что построенная модель полностью отвечает выдвинутым требованиям пользователя.

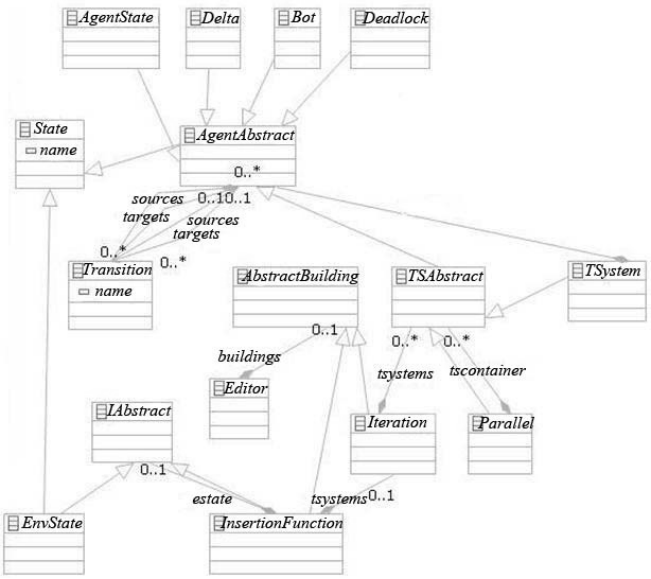

Рис. 2. Доменная модель графического редактора *IMS*

Классы *AgentState*, *Delta*, *Bot*, *Deadlock* реализуют требование 1), наследуя свойства абстрактного класса *AgentAbstract*.

Требованию 2) соответствует класс *Transition*, который находится в отношении 0 к 1 по отношению к элементу, который он соединяет (*AgentAbstract*). *AgentAbstract*, *source*, *targets* означают экземпляры наследников класса *AgentAbstract*, соединяющие класс *Transition*. Соответственно класс *AgentAbstrcut* находится в отношении 0 к ∞, которое означает, что для каждой пары вершин может быть проведено бесконечно много транзакций. Для операции недетерминированного выбора из каждого экземпляра выходит более одной транзакции, а если эта транзакция размечена, тогда это операция префиксинга.

Для удовлетворения требования 3) вводится понятие графической абстрактной транзиционной системы класс *TSAbstract*, представления транзиционной системы – класс *TSystem*, который наследует абстрактный класс *TSAbstract* и понятие параллельной композиции (класс *Parallel*). Этот класс относится к классу *TSAbstract* как 0 к ∞, что реализует также и вложенность параллельной композиции. Последовательная композиция реализуется с помощью класса *Transition*. Отношения между классами *TSAbstract* и *Transition* точно такие же, как и между классами *AgentAbstract* и *Transition*. Вложенность понятий реализуется с помощью класса *Interaction*, который находится в отношении 0 к ∞ с классом *TSAbstract*.

Для удовлетворения требования 4) вводится понятие состояние среды *EnvState* и функции погружения *InsertionFunction*. Для того чтобы удовлетворить вложенность понятий, эти классы наследуют класс *IAbstract*.

Далее для построения самого графического редактора вводится понятие абстрактного построения, которое в редакторе может быть не более одного (класс *AbstructBuilding*), этот класс наследуют два класса *Interaction* и *InsertionFunction*, что позволяет в редакторе использовать две графические модели: функцию погружения или взаимодействие транзиционных систем (класс *Editor*, который находиться в отношении 0 к 1 к классу *Abstract Building*).

После того, как была построена доменная модель графического редактора *IMS*, все необхо-

димые дальнейшие действия можно провести в автоматическом режиме средствами *Eclipse* (или в полуавтоматическом режиме, если сгенерированные свойства моделей надо поменять). В результате описанных действий мы получили графический редактор *IMS*.

Пример диаграммы транзиционной системы

Рассмотрим простой пример транзиционной системы (алгебраическая модель): $((a + c; 0) + ((d + c); (a + b)))$.

Соответствующая ей графическая модель будет выглядеть так:

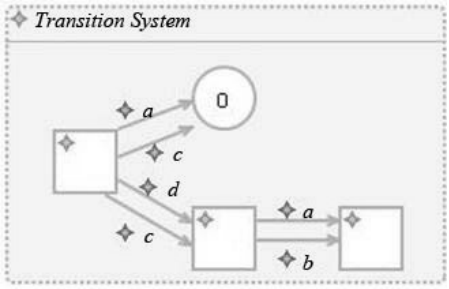

Рис. 3. Пример транзиционной системы в графическом редакторе

Очевидно, что a, b, c, d – действия, о чем свидетельствуют определения алгебры поведений и последовательной композиции. Поскольку состояния агентов не определены, то все квадраты пустые.

Преобразование графической модели в алгебраическую и интерактивный режим

Физически созданная модель представляет собой два файла: данных (**.editor*) и графического представления (**.editor diagram*). Оба файла в формате *XML*. Модель, изображенная на рис. 3, достаточно просто преобразовывается в соответствующую алгебраическую модель: $a.E; 0 + c.E; 0 + (d.E; (a.E + b.E)) + (c.E; (a.E + b.E)) = (a.E + b.E; 0) + ((d.E + c.E); (a.E + b.E))$. Поскольку все состояния агентов пустые, то можно их убрать из алгебраической формы записи, получим: $(a + c; 0) + ((d + c); (a + b))$.

Аналогичные соображения распространяются и на общий случай.

Более сложно обстоит дело с интеграцией такого редактора в существующую систему:

- система должна позволять перехватывать стандартные потоки ввода/вывода;
- система должна обновлять файл графического представления и посылать сообщения редактору о том, что файл надо обновить, когда мы находимся в состоянии определения пути для недетерминированного выбора.

Если удовлетворение первого требования ложится полностью на разработчиков системы, с которой будет интегрироваться редактор, то второе требование удовлетворить оказалось не так просто. Это связано с тем, что редактор и система – два разных потока, и в общем случае *Eclipse* не гарантирует обновление загруженных в него ресурсов. Существует множество решений этой проблемы: послать сообщение (сообщение может не дойти до пользовательского кода в *Eclipse*), использовать системные объекты или некий сервер для обмена данных с этим приложением.

Таким образом, алгоритм взаимодействия *Elipse* и системы, которая использует модели из него, выглядит следующим образом:

Ш а г 1. Запускается *Eclipse* с открытой в нем моделью, модель инициализирует запуск сервера – обработчика сообщений от системы.

Ш а г 2. Настраивается меню запуска приложения, в котором в командной строке системе передается имя модели, с которой мы хотим работать.

Ш а г 3. При запуске система загружает данные модели и далее, при необходимости, изменяет их, посылает соответствующие сообщения в редактор и выводит всю необходимую информацию в консоль.

Ш а г 4. Сервер, получая сообщение об обновлении, инициализирует процедуру обновления ресурсов в *Eclipse*.

Ш а г 5. Пользователь выбирает в консоли *Eclipse* нужную ветку недетерминированного выбора, после чего система возвращает все данные в начальное состояние и посылает сообщение об обновлении ресурсов редактору.

Ш а г 6. Редактор обновляет ресурсы, и система продолжает выполнять программу.

Заключение. Использование технологии *GMF Eclipse* существенно ускоряет процесс создания графических редакторов, которые, кроме задуманной функциональности, получают дос-

таточно большой набор функций *Eclipse* управления построенной моделью. Однако, с другой стороны, использование технологии *GMF Eclipse* привязывает созданный редактор к среде *Eclipse*, что, на взгляд авторов, хуже, чем независимое приложение.

1. *APS & IMS. – http//www. apsystem.org,ua*
2. *Летичевський А.А.*, *Летичевський А.А.* (*мол.*), *Песчаненко В.С.* Оптимізація переписуючої машини системи алгебраїчного програмування *APS* // Вісн. Харк. нац. ун-ту. Сер. Математичне моделювання. Інформаційні технології. Автоматизовані системи управління. – 2009. – № 847. – **11**. – С. 213–220.
3. *http//www.icyb.kiev.ua*
4. *http//riit.ksu.ks.ua*
5. *Letichevsky A.A.*, *Kapitonova Ju.V.*, *Konozenko S.V.* Computations in APS // Theor. Comp. Sci. – 1993. – **119**. – P. 145–171.
6. *http://www.eclipse. org/modeling/gmf/*
7. *http://www.eclipse.org*, свободный
8. *Insertion* programming / A. Letichevsky (Jr.), Ju. Kapitonova, V. Volkov et al. // Kibernetika and System Analysis. – 2003. – **1**. – P. 19–32.
9. *Park D.* Concurrency and automata on infinite sequences, in: LNCS 104. – Berlin: Springer Verlag, 1981. – P. 167–183.
10. *Letichevsky A.* Algebra of behavior transformations and its applications // Structural theory of Automata, Semigroups, and Universal Algebra, NATO Science Series II. Mathematics, Physics and Chemistry. – 2005. – Springer – **207**. – P. 241–272.
11. *http://www.intuit.ru/department/se/vismodtp/*10/3.*html*

Поступила 27.12.2010

Тел. для справок: (095) 324-1557 (Херсон)
E-mail: *vladim@ksu.ks.ua*, *vladimirius@gmail.com*

●

Окончание статьи Р.Ю. Лопаткина и др.

Сервер предоставляет пользователям *web*-сервис по удаленному доступу к экспериментальным установкам и реализуется в виде динамического *web*-сайта с возможностью удаленного проведения экспериментов. Чтобы обеспечить и организовать коллективную работу, на сервере должны быть реализованы механизмы авторизации и аутентификации, планировщик, база данных экспериментов и база методических материалов.

Заключение. Реализованный на основе сетевых технологий УПАК – достаточно гибкая система, позволяющая легко строить различные схемы взаимодействия удаленного пользователя с реальной экспериментальной установкой. Одна из основных целей для дальнейшего развития системы – организация центров коллективного использования удаленных лабораторий, которые были бы востребованы при дистанционном обучении.

1. *Чефранова А.О.* Дистанционное обучение физике в школе и вузе: теоретические аспекты. – М.: Прометей, 2005. – 329 с.
2. *Чефранова А.О.* Дистанционное обучение физике в школе и вузе: практические аспекты. – М.: Прометей, 2006. – 252 с.
3. *Арлоу Д.*, *Нейштадт И. UML* 2 и Унифицированный процесс. Практический объектно-ориентированный анализ и проектирование. – СПб: Символ-Плюс, 2007. – 624 с.
4. *http://ru.wikipedia.org/wiki/Ethernet*
5. *IEEE* Standard for Information technology. – *http://standards.ieee.org/about/get/*802/802.3.*html*

Поступила 24.06.2011

Тел. для справок: (0542) 36-2109 (Сумы)
E-mail: *l_rom@mail.ru*, *k_vic@mail.ru*, *va.ivashchenko@gmail.com*, *ignatenko_sergey@inbox.ru*

●