УДК 681.001.51

МАТЕМАТИЧЕСКАЯ ФОРМАЛИЗАЦИЯ ТЕСТОВОЙ МОДЕЛИ ВЗАИМОДЕЙСТВИЯ ОБЪЕКТОВ СЛОЖНОЙ ПРОГРАММНОЙ СИСТЕМЫ

С.А. Олизаренко
(представил д.т.н., проф. Ю.Г. Даник)

Рассмотрены вопросы формализации процессов тестирования объектно-ориентированных моделей сложных программных систем на начальных стадиях их жизненных циклов.

В настоящее время, одной из основных особенностей процесса создания сложных программных систем (СПС) является особенность, связанная с концентрацией сложности на стадиях анализа и проектирования СПС при относительно невысокой сложности и трудоемкости последующих стадий жизненного цикла СПС (ЖЦ СПС) [3, 5]. Данная особенность определяет доминирование начальных стадий ЖЦ СПС с точки зрения внесения различных ошибок в проектные решения при создании СПС. При этом процессы анализа и проектирования СПС определяются, прежде всего, как процессы построения и последовательного развития комплекса согласованных моделей в рамках так называемой интегрированной модели СПС (ИМ СПС).

Из множества языков, используемых для моделирования СПС, наиболее эффективным для описания проектных решений является универсальный язык объектно-ориентированного моделирования UML [1 – 3]. UML предлагает достаточно богатый набор моделей, используемых, как правило, на стадиях анализа и проектирования, при этом в нём собраны общепризнанные концепции практически из всех существующих объектно-ориентированных методов [1, 2].

В связи с тем, что стадии анализа и проектирования являются доминирующими с точки зрения внесения ошибок, в данной работе тестирование СПС предлагается рассматривать не как обособленный вид деятельности или одну стадию ЖЦ СПС, в которой выполняется оценка качества конечного продукта, а как возможность оценки качества на протяжении всего ЖЦ СПС с постоянной возможностью организации обратной связи, когда еще есть время и ресурсы для принятия необходимых мер по устранению допущенных ошибок. Другими словами, предлагается производить тестирование на различных стадиях создания СПС с проверкой соответствующих целевых объектов тестирования – от моделей на начальных стадиях до завершенных программных приложений на стадиях реализации и сопровождения.

В общем случае порядок тестирования моделей в нотации UML, полученных на стадиях анализа и проектирования, предлагается осуществлять по обобщенной методике, предложенной в [6]:

1. Определение пространства тестирования – определение области и глубины проверки. Область определяется набором случаев использования (сценариями). Глубина проверки определяется путем описания охватываемого уровня детализации (стадия анализа или проектирования, концептуальное или логическое моделирование).

2. На основе моделей UML построение MUT-моделей (Model Under Test – тестируемая модель), например, с использованием средств имитационного моделирования [7].

3. Разработка тестовых случаев. Разрабатываются входные данные тестовых случаев, которые определяются из соответствующих прецедентов (или точнее сценариев) и проектной документации. Определяются ожидаемые результаты тестирования.

4. Установление критериев для измерения покрытия MUT-моделей. Определяются элементы моделей, необходимые для покрытия тестовыми случаями: классы, отношения, объекты и/или сообщения (события).

5. Применение тестовых случаев к тестируемой модели. На вход MUT-моделей подаются наборы данных, предусмотренные спецификацией тестового случая, и фиксируются результаты их обработки, которые затем сравниваются с ожидаемыми результатами, указанными в тестовом случае.

6. Сбор и анализ результатов прогона тестовых случаев:

– формирование статистических данных в виде процентных отношений удачных и неудачных исходов тестирования;

– разбиение результатов тестирования по категориям (по типам тестируемых элементов моделей);

– формирование каталогов ошибок и отчетов с описанием ошибок;

– оценка результатов тестирования с точки зрения корректности, полноты и непротиворечивости MUT-моделей.

Особое место в рамках тестирования проектных решений занимает тестирование поведенческих свойств UML-моделей, так как оно позволяет оценить такие основные параметры качества разрабатываемого программного продукта уже на начальных стадиях ЖЦ СПС как [4]:

– безотказность – в данной интерпретации отсутствие “зависаний”, аварийных отказов и т.д.;

– функциональная полнота – в данной интерпретации реализация требуемых прецедентов или ожидаемого поведения;

– реактивность – в данной интерпретации возможность своевременного реагирования на предопределенные события.

С целью формализации и последующей автоматизации тестирования модели кооперации объектов (МКО) в нотации UML предлагается метод формализованного представления тестовой модели взаимодействия объектов сложной программной системы. Разработанный метод базируется

на основных положениях объектно-ориентированной методологии, имитационного моделирования и теории сетей Петри. Метод включает следующие взаимосвязанные составляющие:

– формализованное представление МКО в нотации UML в терминах формального аппарата Е-сетей;

– алгоритм функционирования перехода Е-сетевой тестовой модели взаимодействия объектов;

– процедуру формализации результатов функционирования тестовой модели;

– процедуру контроля результатов функционирования тестовой модели.

Рассмотрим математическую формализацию МКО в нотации UML в терминах формального аппарата Е-сетей более детально.

Кооперация – это описание нескольких объектов, которые взаимодействуют между собой, для реализации определенного поведения [1]. То есть, МКО акцентирует внимание на организации объектов, принимающих участие во взаимодействии. При этом модель кооперации MCO_{su} будем задавать следующей совокупностью:

1) множеством объектов $SO_{su} = \{so_1, so_2, \ldots, so_n\}$, реализующих прецедент su, где n – мощность множества SO_{su};

2) множеством актеров ST_{su};

3) множеством экземпляров событий SL_{su};

4) множеством деятельностей SH_{su};

5) множеством связей SR_{su};

6) функцией порождения F_{SotSl}: $(SO_{su} \cup ST_{su}) \times SL_{su} \to \{0, 1\}$, если $F_{\mathrm{SotSl}}(sot, sl) = 1$, то сигнал управления $sl \in SL_{su}$ порождается объектом или внешней сущностью sot, где $sot \in (SO_{su} \cup ST_{su})$;

7) отображением F_{SrSo}: $SR_{su} \to SO_{su}$, которое определяет возможные связи между объектами.

Определения класса, объекта, события, актера, связи введены в [1, 2].

В общем случае Е-сеть задается совокупностью следующих множеств $N_E = (B^E, B_P{}^E, B_R{}^E, Z^E, F_{BZ}{}^E, F_{ZB}{}^E, M_0{}^E)$, где $B^E = \{b_1, \ldots, b_m\}$ – конечное множество позиций, $B^E \neq \varnothing$; $B_P{}^E = \{b_i, \ldots, b_l\}$ – конечное множество периферийных позиций, $B_P{}^E \subset B$; $B_R{}^E = \{b_j, \ldots, b_k\}$ – конечное множество решающих позиций, $B_R{}^E \subset B^E$; $Z^E = \{z_1, \ldots, z_f\}$ – конечное множество описаний переходов, $Z^E \neq \varnothing$, $z_i = (\beta, t(z_i), \rho(z_i))$ (здесь β – тип перехода; $t(z_i)$ – время перехода; $\zeta(z_i)$ – процедура перехода); $F_{BZ}{}^E$ – прямая функция инцидентности, $F_{BZ}{}^E$: $B^E \times Z^E \to \{0, 1\}$; $F_{ZB}{}^E$ – обратная функция инцидентности, $F_{ZB}{}^E$: $Z^E \times B^E \to \{0, 1\}$; $M_0{}^E$ – начальная маркировка сети, $M_0{}^E$: $B^E \to \{0, 1\}$.

Таким образом, $MCO_{su} = (SO_{su}, ST_{su}, SL_{su}, SH_{su}, SR_{su}, F_{SotSl}, F_{SrSo})$ преобразуется в соответствующую Е–сеть $N_E{}^{MCO}$ следующим образом:

1. Каждому элементу множества объектов SO_{su} или множества внешних сущностей (актеров) ST_{su} соответствует единственный элемент множества Z^E описаний переходов Е-сети, т.е. определено биективное отображение

$$F_{ZSot}: Z^E \to (SO_{su} \cup ST_{su}).$$

2. Если позицию сети рассматривать как некоторое событие ИМ СПС, то метку, поступающую в данную позицию, можно рассматривать как экземпляр события из множества SL_{su}. Так как структурно Е-сети относятся к подклассу маркированных графов, т.е. $\forall b \in B^E/B_P{}^E\ F_{BZ}{}^E(b, z) = F_{ZB}{}^E(z, b) = 1$, то некоторому событию ИМ СПС может соответствовать несколько элементов множества B^E (по количеству объектов и (или) внешних сущностей, порождающих экземпляры данного события).

В связи с тем, что объект или актер в общем случае может принимать в ходе своего жизненного цикла более одного экземпляра события от одного источника, то для представления события ИМ СПС предлагается использовать позиции-очереди. Обозначим множество позиций-очередей через $B_Q{}^E$, где $B_Q{}^E \subseteq B^E/B_P{}^E$, $B_Q{}^E \cap B_R{}^E = \varnothing$. При этом принцип действия позиции-очереди в нашем рассмотрении состоит в том, что в нее может помещаться несколько меток (накапливаются экземпляры события), и метка, пришедшая первой, первой из нее выходит.

Соответствие элемента множества событий ИМ СПС одному или нескольким элементам множества $B_Q{}^E$ можно определить как сюръективное отображение

$$F_{BSe}: B_Q{}^E \to SE_{su},$$

где $\forall se \in SE_{su}(\exists sl \in SL_{su})$.

Так как экземпляр события порождается объектом или внешней сущностью, то в нашем рассмотрении только решающие позиции являются периферийными, т.е. $B_P{}^E = B_R{}^E$.

3. Прямая и обратная функции инцидентности определяются по формулам:

$$F_{BZ}{}^E(b, z) = \begin{cases} 1, \text{ если выполняется условие, } \exists st \in ST_{su}\ F_{BSe}(b) = se, F_{ZSot}(z) = st \\ \text{или } \exists so \in SO_{su}\ F_{BSe}(b) = se, F_{ZSot}(z) = so; \\ 0, \text{ если условие не выполняется;} \end{cases}$$

$$F_{ZB}{}^E(z, b) = \begin{cases} 1, \text{ если выполняется условие, } \exists sot \in (SO_{su} \cup ST_{su}), \\ F_{SotSl}(sot, sl) = 1, F_{BSe}(b) = se, F_{SlSe}(sl) = se, F_{ZSot}(z) = so; \\ 0, \text{ если условие не выполняется.} \end{cases}$$

4. Начальная маркировка сети определяется по формуле

$$M_0(b) = \left| \cup_{sl \in SLsu} \{F_{Se\sigma(Sa)}(F_{SlSe}(sl))\} \right|,$$

где $\forall b \in B_Q{}^E(F_{BSe}(b) = se, F_{SlSe}: SL \to SE, F_{SlSe}(sl) = se, F_{SotSl}(sot, sl) = 1)$, а

$\forall b \in B_P^E(M_0^E(b) = \varnothing)$, $F_{Se\sigma(Sa)}$: $SE \to \sigma(SA)$, SA – множество атрибутов ИМ СПС.

Важной особенностью Е-сети является детализация представления метки. С каждой меткой в Е-сети связаны n описателей. В нашем рассмотрении в зависимости от позиции-очереди, в которую попадает метка, и будем определять набор ее описателей, при этом в дальнейшем будем считать, что с каждой меткой обязательно связан описатель, в котором будет задаваться время (очередность) поступления метки в позицию-очередь.

Таким образом, результат соответствия описателей некоторой метки j в позиции-очереди $b \in B_Q^E$ данным некоторого экземпляра события $sl \in SL_{su}$ можно формализовать как инъективное отображение:

$$F_{bSa}: \{b^j[i]\} \to F_{Se\sigma(Sa)}(F_{SlSe}(sl)),$$

где $b^j[i]$ – i-й описатель некоторой метки j в позиции-очереди b; $F_{Se\sigma(Sa)}(F_{SlSe}(sl))$ – множество данных экземпляра события $sl \in SL_{su}$ и $F_{BSe}(b) = se$, $F_{SlSe}(sl) = se$; $i = 1, \ldots, n$; n – количество описателей меток в позиции-очереди b; $j = 1, \ldots, m$; m – количество меток в позиции-очереди b, т.е. поступивших экземпляров одного события от одной сущности.

Для обеспечения формального единства МКО и ее сетевой тестовой модели предлагается ввести набор описателей перехода z, соответствующий набору означенных атрибутов объекта $so \in SO_{su}$. Результат данного соответствия можно формализовать как инъективное отображение:

$$F_{zSa}: \{z[i]\} \to F_{Sc\sigma(Sa)}(F_{SoSc}(so)),$$

где $z[i]$ – i-й описатель перехода z, $F_{ZSot}(z) = so$, $i = 1, \ldots, n$; n – количество описателей перехода z. При этом в дальнейшем будем считать, что $\forall z \in Z^E(F_{za}(z[1]) = a_S)$, где a_S – атрибут состояния, $F_{SoSc}(so) = sc$, F_{SoSc}: $SO \to SC$, SC – множество классов ИМ СПС.

В общем случае Е-сети являются безопасными, т.е. $\forall b \in B^E$ $(M^E(b) \le 1)$. При этом выполнение данного условия поддерживается в сети искусственно за счет изменения логики работы перехода (условия возбуждения перехода).

В общем случае условие возбуждения перехода представляется как:

$$\forall z \in Z^E(\forall b \in F_{BZ}{}^{\beta}(z) \to (M^E(b) - F_{BZ}{}^{\beta}(b, z) = 0) \;\&$$
$$\forall b \in F_{ZB}{}^{\beta}(z) \to (M^E(b) - F_{ZB}{}^{\beta}(z, b) = M^E(b))),$$

где $F_{BZ}{}^{\beta}(z) \subseteq F_{BZ}(z)$; $F_{ZB}{}^{\beta}(z) \subseteq F_{ZB}$; $b \in B^E$; а состав подмножеств $F_{BZ}{}^{\beta}$ и $F_{ZB}{}^{\beta}$ зависит от типа Е-сетевого перехода β.

Однако, так как в нашем рассмотрении используются только позиции-очереди и решающие позиции, причем последние являются только периферийными, то условие возбуждения перехода будем представлять как

$$\forall z \in Z^E(\exists b \in F_{BZ}{}^{\beta}(z) \to (M^E(b) \ge 1) \;\&\; \forall b \in F_{ZB}{}^{\beta}(z) \to (M^E(b) \ge 0)),$$

где $F_{BZ}^{\beta} \subseteq F_{BZ}$; $F_{ZB}^{\beta} \subseteq F_{ZB}$; $b \in B_Q^E$. При этом в состав подмножества F_{BZ}^{β} в нашем рассмотрении входит только одна позиция (переход из одного состояния в другое осуществляется посредством одного события (экземпляра события)).

Исходя из вышесказанного можно сделать вывод, что предлагаемый формальный подход предусматривает новое определение Е-сетевого перехода, а именно:

– переход сети рассматривается в виде объекта в нотации UML с относящимися к нему позициями-очередями в виде событий класса, экземпляром которого является заданный объект;

– метка, поступившая в некоторую позицию, рассматривается как экземпляр события класса;

– в зависимости от позиции-очереди, в которую попадает метка, определяется набор ее описателей, при этом с каждой меткой обязательно связан описатель, в котором задается время (очередность) поступления метки в позицию-очередь;

– для обеспечения формального единства МКО и ее сетевой модели вводится набор описателей и для перехода, соответствующий набору означенных атрибутов объекта.

Таким образом, реализация метода представления МКО в рамках предложенного формализованного подхода позволяет автоматизировать процесс контроля показателей качества, характеризующих различные свойства создаваемой СПС, уже на начальных стадиях ЖЦ СПС до стадии программной реализации.

ЛИТЕРАТУРА

1. *Буч Г., Рамбо Дж., Джекобсон А. UML. Руководство пользователя: Пер. с англ. – М.: ДМК, 2001. – 432 с.*
2. *Буч Г., Рамбо Дж., Джекобсон А. UML. Специальный справочник: Пер. с англ. – СПб.: Питер, 2002. – 656 с.*
3. *Вендров А.М. Проектирование программного обеспечения экономических информационных систем. – М.: Финансы и статистика, 2000. – 347 с.*
4. *ДСТУ 2850 – 94 ПЗ ЄОМ. Показники та методи оцінки якості.– К.: Держстандарт України, 1994. – 20 с.*
5. *Калянов Г.Н. CASE–технологии: консалтинг в автоматизации бизнес-процессов. – М.: Горячая линия-Телеком, 2000. – 320 с.*
6. *Макгрегор Д., Сайкс Д. Тестирование объектно-ориентированного программного обеспечения. Практическое пособие: Пер. с англ. – К.: ООО «ТИД ДС», 2002. – 432 с.*
7. *Технология системного моделирования / Е.Ф. Аврамчук, А.А. Вавилов, С.В. Емельянов и др. – М.: Машиностроение; Берлин: Техник, 1988. – 520 с.*

Поступила 15.10.2002

***ОЛИЗАРЕНКО Сергей Анатольевич**, старший научный сотрудник научного цен-*

тра при ХВУ. Окончил ХВВКИУРВ в 1993 году. Область научных интересов – системы обработки информации.