СБОРНИК **НАУЧНЫХ ТРУДОВ НГТУ. – 2016. – № 2(84).** – 99–106

СОВРЕМЕННЫЕ ИНФОРМАЦИОННЫЕ ТЕХНОЛОГИИ

УДК 004.434:004.94

ВЫЯВЛЕНИЕ КЛАССОВ UML-МОДЕЛИ С ПОМОЩЬЮ ДОПОЛНЕННОЙ ДИАГРАММЫ ДЕЯТЕЛЬНОСТИ*

А.В. АНИКИН[1], И.Р. НАЗАРОВ[2]

[1] *630073, РФ, г. Новосибирск, пр. Карла Маркса, 20, Новосибирский государственный технический университет, магистрант кафедры автоматики. E-mail: anikantonsd@yandex.ru*

[2] *630073, РФ, г. Новосибирск, пр. Карла Маркса, 20, Новосибирский государственный технический университет, магистрант кафедры автоматики. E-mail: igorfresh@mail.ru*

UML (Unified Modeling Language – унифицированный язык моделирования) – язык для объектного моделирования в области разработки программного обеспечения, системного проектирования, моделирования бизнес-процессов и документирования с помощью графического описания. UML является языком широкого профиля, это открытый стандарт, использующий графические обозначения для создания абстрактной модели системы, называемой UML-моделью. UML не является языком программирования, но на основании UML-моделей возможна генерация кода. Модель класса является основным элементом для описания внутренней статической структуры программной системы, создания модели данных предметной области и ее последующей программной реализации в виде взаимодействующих экземпляров классов или таблиц реляционной модели. Классы, их свойства и методы, а также отношения между ними в UML представляются диаграммой классов (Class diagram). При проектировании или описании какой-либо системы возникает вопрос о том, какие классы должны быть представлены на этой диаграмме. С учетом того, что простых алгоритмов или универсальных подходов не существует, выявление классов, их семантических связей, кратности этих связей является довольно трудной задачей, которую, однако, необходимо решить в процессе создания моделей. В данной статье предлагается способ, облегчающий эту задачу путем дополнения диаграммы активности, входящей в стандартную нотации UML, дополнительными данными.

Ключевые слова: UML, моделирование, модель системы, диаграмма классов, класс, выявление классов, диаграмма деятельности, входящая/исходящая информация

DOI: 10.17212/2307-6879-2016-2-99-106

---

* Статья получена 4 марта 2016 г.

ВВЕДЕНИЕ

В литературе [1–14] рассматриваются распространенные способы выявления классов для создания модели. Широко применяются такие подходы:

- метод «Существительное/Глагол»: анализируется текст доступной документации, существительные и именные группы указывают на классы или атрибуты. Глаголы и глагольные группы служат признаком обязанностей или операций;
- метод CRC-анализа (техника мозгового штурма);
- применение стереотипов RUP (поиск классов, которые можно обозначить стереотипами boundary (граница), control (управление) и entity (сущность));
- применение готовых шаблонов классов анализа.

Необходимо подчеркнуть, что не существует простого алгоритма или универсального подхода, дающего гарантированный результат [5]. В перечисленных подходах решение основывается на возможно неполном представлении конкретного разработчика о предметной области и моделируемой системе и отсутствуют предложения о том, как формализовать задачу выявления классов и тем самым обеспечить ее решение [15].

1. ОПИСАНИЕ МЕТОДА

В данной статье для выявления классов модели предполагается использовать модель поведения проектируемой системы (программы) в виде алгоритмической структуры, основанной на представлении разработчика о предметной области. Средством такого представления в UML служит диаграмма деятельности. Наряду с диаграммой состояний (State Machine diagram) – конечных автоматов диаграмма деятельности (Activity diagram) является одной из диаграмм, описывающих поведение системы, но в отличие от диаграммы состояний, в которой описываются состояния системы и ее частей, переходы между ними и условия их выполнения, в диаграмме деятельности детализируются особенности алгоритмической и процедурной реализации выполняемых системой операций, т. е. внимание уделяется последовательному и параллельному выполнению отдельных действий, соединенных между собой потоками, которые идут от выходов одного действия ко входу другого.

Для выявления классов предлагается дополнить диаграмму деятельности описанием входящей и исходящей информации, которая должна обрабатываться при исполнении каждого действия алгоритма на специальной дорожке. Эта информация представляется в виде потока объектов (object flow), связан-

ных с действиями алгоритма, причем эти действия могут изменять их состояние. Таким образом, появляется возможность выявить объекты, которые необходимы для исполнения алгоритма и являются формальным следствием любого его действия. В качестве таких объектов по мере исполнения алгоритмических действий появляются объекты для хранения информации, а действия создают их новые экземпляры либо меняют их свойства и состояние. Наличие таких объектов свидетельствует о необходимости создания соответствующего класса сущности. Граничные классы выявляются в виде объектов, появляющихся на этой дорожке, например, при действиях алгоритма, связанных с участием пользователя для получения информации от объекта-сущности.

2. ПРИМЕРЫ

На рис. 1 изображена простейшая, дополненная информацией о входах и выходах действий диаграмма деятельности, описывающая последовательность действий при взаимодействии водителя и автоматического шлагбаума.

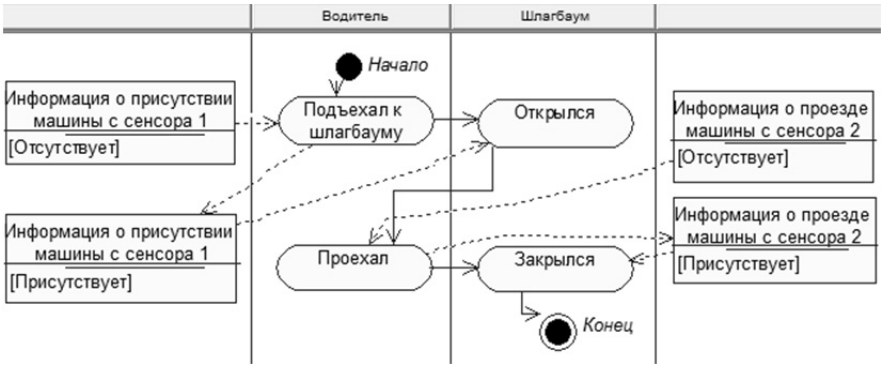

Рис. 1. Диаграмма деятельности шлагбаума

На плавательных дорожках (swimlanes) «Водитель» и «Шлагбаум» располагаются выполняемые ими действия соответственно. При подъезде машины информацию об этом получает соответствующий сенсор и передает ее на шлагбаум, который в результате открывается. Аналогично при получении информации о проезде вторым сенсором и передаче ее на шлагбаум он закрывается. Исходя из информации о входах и выходах действий представленной диаграммы, можно сделать вывод о необходимости класса сущности «Сен-

сор», который меняет свое состояние в зависимости от действий, указанных в алгоритме.

Рассмотрим другой пример: описание поведения системы при проектировании базы данных. На диаграмме деятельности изображено взаимодействие между пользователем и системой поиска товаров в виде описания их действий на соответствующих плавательных дорожках и информации о вводе/выводе. Пользователь активирует поисковую систему, система требует ввести данные о требуемом товаре в форму поиска. Пользователь вводит данные о товаре, на основании которых система производит поиск среди имеющихся вариантов. В результате поиска появляется информация о списке подходящих товаров, которая в дальнейшем выводится системой. Исходя из представленной диаграммы следует, что для выполнения поиска необходимы класс сущности «Товар» и граничный класс «Форма поиска».

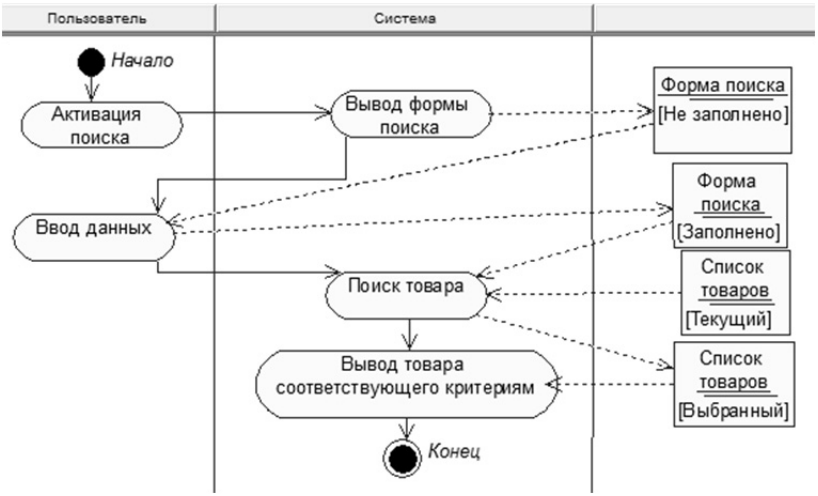

Рис. 2. Диаграмма деятельности доступа к базе данных

Таким образом, дополненная диаграмма активности позволяет получить более глубокое понимание работы системы и может быть использована при разработке диаграммы классов системы наряду с другими способами выявления классов.

ЗАКЛЮЧЕНИЕ

В данной работе представлена методика дополнения диаграммы деятельности, обеспечивающая большую информативность при проверке создаваемого алгоритма за счет использования потока объектов как входной и выходной информации действий алгоритма поведения системы. Использование информации о входных и выходных объектах позволяет также выявить классы в качестве результата формальных действий моделируемого алгоритма.

СПИСОК ЛИТЕРАТУРЫ

1. *Арлоу Д., Нейштадт И.* UML 2 и Унифицированный процесс: практический объектно-ориентированный анализ и проектирование. – 2-е изд. – СПб.: Символ, 2007. – 624 с.

2. *Буч Г., Рамбо Д. Джекобсон А.* Язык UML. Руководство пользователя. – М.: ДМК, 2000. – 432 с.

3. *Фаулер М.* UML. Основы. – 3-е изд. – СПб.: Символ, 2004. – 192 с.

4. *Ларман К.* Применение UML и шаблонов проектирования. – 2-е изд. – М.: Вильямс, 2004. – 624 с.

5. *Рамбо Дж., Блаха М.* UML 2.0. Объектно-ориентированное моделирование и разработка. – СПб.: Питер, 2007. – 544 с.

6. *Douglass B.* Real-time UML workshop for embedded systems. – Oxford, UK: Newnes, 2006. – 576 p.

7. *Arlow J., Neustadt I.* Enterprise patterns and MDA: building better software with archetype patterns and UML. – Boston, USA: Addison Wesley, 2003. – 528 p.

8. *Hamilton K., Miles R.* Learning UML 2.0. – Sebastopol, USA: O'Reilly, 2006. – 286 p.

9. *Кватрани Т.* Rational Rose 2000 и UML. Визуальное моделирование. – М.: ДМК, 2001. – 176 с.

10. *Rosenberg D., Stephen M.* Use case driven object modeling with UML. Theory and practice. – New York, USA: Apress, 2007. – 438 p.

11. *Мацяшек Л.* Анализ и проектирование информационных систем с помощью UML 2.0. – М.: Вильямс, 2008. – 816 с.

12. *Douglass B.* Real-time design patterns. – Oxford, UK: Newnes, 2002. – 528 p.

13. *Douglass B.* Systems engineering with SysML. – Oxford, UK: Newnes, 2006. – 576 p.

14. *Иванов Д., Новиков Ф.* Моделирование на UML. – СПб.: СПбГУ ИТМО, 2010. – 200 с.

15. *Meyer B., Hall P.* Object oriented software construction. – Upper Saddle River, USA: Prentice-Hall, 1997. – 1255 p.

Аникин Антон Викторович, магистрант кафедры автоматики Новосибирского государственного технического университета по направлению «Управление в технических системах». Основное направление научных исследований – автоматизированное проектирование программных систем. E-mail: anikantonsd@yandex.ru

Назаров Игорь Романович, магистрант кафедры автоматики Новосибирского государственного технического университета по направлению «Управление в технических системах». Основное направление научных исследований – автоматизированное проектирование программных систем. E-mail: igorfresh@mail.ru

Class identification of UML-model using extended activity diagram*

A.V. Anikin[1], I.R. Nazarov[2]

[1] *Novosibirsk State Technical University, 20 Karl Marks Avenue, Novosibirsk, 630073, Russian Federation, undergraduate student of the automation department. E-mail: anikantonsd@yandex.ru*

[2] *Novosibirsk State Technical University, 20 Karl Marks Avenue, Novosibirsk, 630073, Russian Federation, undergraduate student of the automation department. E-mail: igorfresh@mail.ru*

UML (Unified Modeling Language) – a language for object modeling in software development, system design, business process modeling and visualized documenting. UML is a general-purpose language, it is an open standard that uses figures for creating an abstract model of the system, called the UML-model. UML is not a programming language, but code generation is available for the UML-based models. Class model is a key element to describe the static structure of internal software system, to create a data model of domain and its further implementation of the program in the form of interacting instances of classes or tables of the relational model. The classes, their properties, methods and the relationships among them are represented in the UML class diagram. There is a question about what classes should be represented in this diagram during design or specification of any system. Given simple algorithms or universal approaches do not exist, identifying classes and their semantic relations, the multiplicity of these connections is quite a difficult task, which, however, have to be solved in the process of creating mod-

* *Received 04 March 2016.*

els. This article provides a method that facilitates this task by enhancing the activity diagram, included in the standard UML notation, with the additional data.

Keywords: UML, modeling, model of the system, class diagram, class, class identification, activity diagram, incoming / outgoing information

DOI: 10.17212/2307-6879-2016-2-99-106

REFERENCES

1. Arlow D., Neustadt I. *UML 2 and the unified process: practical object-oriented analysis and design*. 2nd ed. Boston, USA, Addison Wesley, 2005. 624 p. (Russ. ed.: Arlou D., Neishtadt I. *UML 2 i Unifitsirovannyi protsess: prakticheskii ob"ektno-orientirovannyi analiz i proektirovanie*. 2nd ed. St. Petersburg, Simvol Publ., 2007. 624 p.).

2. Booch G., Jacobson I., Rumbaugh J. *The Unified Modeling Language user guide.* Boston, USA, Addison Wesley, 1998. 512 p. (Russ. ed.: Buch G., Rambo D. Dzhekobson A. *Yazyk UML. Rukovodstvo pol'zovatelya.* Moscow, DMK Publ., 2000. 432 p.).

3. Fowler M. *UML distilled.* Boston, USA, Addison Wesley, 2004. 175 p. (Russ. ed.: Fauler M. *UML. Osnovy.* 3rd ed. St. Petersburg, Simvol Publ., 2004. 192 p.).

4. Larman C. *Applying UML and patterns*. 2nd ed. Boston, USA, Addison Wesley, 2004. 736 p. (Russ. ed.: Larman K. *Primenenie UML i shablonov proektirovaniya*. 2nd ed. Moscow, Williams Publ., 2004. 624 p.).

5. Rumbaugh J., Blaha M. *Object-oriented modeling and design with UML.* London, UK, Pearson, 2004. 496 p. (Russ. ed.: Rambo Dzh., Blaha M. *UML 2.0. Ob"ektno-orientirovannoe modelirovanie i razrabotka.* St. Petersburg, Piter Publ., 2007. 544 p.).

6. Douglass B. *Real-time UML workshop for embedded systems.* Oxford, UK, Newnes, 2006. 576 p.

7. Arlow J., Neustadt I. *Enterprise patterns and MDA: building better software with archetype patterns and UML.* Boston, USA, Addison Wesley, 2003. 528 p.

8. Hamilton K., Miles R. *Learning UML 2.0.* Sebastopol, USA, O'Reilly, 2006. 286 p.

9. Kvatrani T. *Rational Rose 2000 i UML. Vizual'noe modelirovanie* [Rational Rose 2000 and UML. Visual modeling]. Moscow, DMK Publ., 2001. 176 p.

10. Rosenberg D., Stephen M. *Use case driven object modeling with UML. Theory and practice.* New York, USA, Apress, 2007. 438 p.

11. Maciaszek L. *Requirements analysis and systems desig*n. Ontario, Canada, Pearson Education Canada, 2007. 656 p. (Russ. ed.: Matsyashek L. *Analiz i proek-*

tirovanie informatsionnykh sistem s pomoshch'yu UML 2.0. Moscow, Williams Publ., 2008. 816 p.).

12. Douglass B. *Real-time design patterns.* Oxford, UK, Newnes, 2002. 528 p.

13. Douglass B. *Systems engineering with SysML.* Oxford, UK, Newnes, 2006. 576 p.

14. Ivanov D., Novikov F. *Modelirovanie na UML* [UML modeling]. St. Petersburg, SPbSU ITMO Publ., 2010. 200 p.

15. Meyer B., Hall P. *Object oriented software construction.* Upper Saddle River, USA, Prentice-Hall, 1997. 1255 p.