УДК 004.031.6

В.В. ЛИТВИНОВ[*], И.В. БОГДАН[**]

ФОРМАЛЬНАЯ ВЕРИФИКАЦИЯ ДИАГРАММЫ КЛАССОВ

[*]Черниговский государственный технологический университет, Чернигов, Украина
[**]Черниговский государственный технологический университет, Чернигов, Украина

Анотація. *У статті описуються найбільш популярні серед існуючих підходів до проведення верифікації UML-діаграми, що використовується частіше за все – діаграми класів. Вказується, що дані методи дають можливість оцінити її коректність лише в окремих аспектах, і найбільш ефективним являється комплексне використання даних методів.*
Ключові слова: *верифікація, шаблон, UML-діаграма, модель, коректність, діаграма класів.*

Аннотация. *В статье описываются наиболее популярные из существующих подходов к верификации наиболее часто используемой UML-диаграммы – диаграммы классов. Указывается, что данные методы позволяют оценить ее корректность только в отдельных аспектах, и наиболее эффективным является комплексное применение данных методов.*
Ключевые слова: *верификация, шаблон, UML-диаграмма, модель, корректность, диаграмма классов.*

Abstract. *The article describes the most popular of the existing approaches to verification of the most frequently used UML-diagram – the class diagram. It is indicated that these methods allow us to estimate its correctness only in certain aspects and integrated application of these methods is considered to be the most effective.*
Keywords: *verification, pattern, UML-diagram, model, correctness, class diagram.*

1. Введение

Формальная верификация программного обеспечения – это приемы и методы формального доказательства (или опровержения) того, что модель программного обеспечения удовлетворяет заданной формальной спецификации [1]. Для того, чтобы доказать формально какое-либо утверждение относительно работы реальной программы, анализируемое программное обеспечение должно быть представлено формальной моделью. Формальная модель обычно проще самой проверяемой программы. Это абстракция, в которой отражены наиболее существенные характеристики программного обеспечения. Таким образом, процесс создания любого программного обеспечения начинается с создания его модели, которой для эффективного дальнейшего использования необходимо пройти формальную верификацию.

Диаграмма классов является одной из основных составляющих модели любого программного обеспечения [2]. Именно поэтому особенно важное значение имеет верификация диаграмм классов. На данный момент существует достаточно большое количество различных подходов к верификации диаграмм классов. Большинство из них позволяют оценить корректность диаграммы классов с различной точки зрения.

2. Построение драйвера

Так, наиболее простым является подход, связанный с построением драйвера [3]. Данный подход позволяет формально оценить правильность создания диаграммы классов и выявить наиболее характерные ошибки. Прежде чем приступать к созданию драйвера, необходимо определить, следует ли выполнять верификацию каждого конкретного класса в автономном режиме, представляя его как отдельный модуль, или же необходимо восприни-

ISSN 1028-9763. Математичні машини і системи, 2013, № 2

мать класс как более крупный компонент системы. Решение принимается на основании следующих факторов:

– роль данного класса в системе, в частности, степень связанного с ним риска;

– сложность класса, измеряемая количеством состояний, операций и связей с другими классами;

– объем трудозатрат, связанных с разработкой драйвера для верификации класса.

Если какой-либо класс должен стать частью некоторой библиотеки классов, то наиболее оптимальной является всесторонняя верификация классов, причем, даже в том случае, если затраты на разработку драйвера окажутся высокими, поскольку очень важным является его корректное функционирование.

Верификация классов из диаграммы классов чаще всего выполняется путем разработки драйвера, который создает экземпляры каждого класса и окружает эти классы соответствующей средой. Таким образом, становится возможным выполнение драйвера. Драйвер посылает одно или большее количество сообщений экземпляру класса в соответствии со спецификацией тестового случая, затем проверяет исход этих сообщений на основании значений ответа, изменения экземпляра и (или) один или большее число параметров сообщения. В обязанности драйвера чаще всего входит удаление любого созданного им экземпляра в том случае, если в языке программирования, таком как C++, имеет место управляемое программистом распределение памяти.

Если для конкретного класса характерны статические элементы данных и (или) операции, то для них также необходимо выполнять верификацию. Такие элементы данных и методы принадлежат самому классу, но не каждому экземпляру этого класса. Класс можно рассматривать как объект. Например, в Java – это экземпляр класса Class.

Если поведение экземпляров класса базируется на значениях атрибутов уровня класса, то все случаи, предназначенные для верификации этих атрибутов уровня класса, должны рассматриваться как расширение состояний этих экземпляров.

Если между двумя и более классами на диаграмме классов присутствует связь «наследование» (обобщение), то драйверу необходимо проверить отсутствие множественного наследования для языка Java, а также отсутствие двунаправленного наследования, при котором каждый из двух классов является и родительским, и дочерним одновременно для всех объектно-ориентированных языков программирования.

Если между классом и интерфейсом на диаграмме классов присутствует связь «реализация», то драйверу необходимо проверить, реализует ли класс все те методы, сигнатуры которых указаны в интерфейсе.

Если в классе присутствует хотя бы один абстрактный метод, то драйверу необходимо проверить, чтобы и класс был абстрактным.

3. Метод шаблонов

Еще одним подходом к верификации диаграммы классов является метод шаблонов. Данный метод предложила в 2004 году Мира Балабан [4]. Понятие шаблона в данном подходе отличается от хорошо известного и понятного понятия шаблона проектирования, который предоставляет собой формализованное описание часто встречающейся задачи проектирования, удачное решение данной задачи, а также рекомендации по применению этого решения в различных ситуациях. Шаблон, согласно методу Мира Балабан, напротив, является примером ошибок при построении диаграммы классов.

Mira Balaban предложила несколько шаблонов.

Один из самых популярных шаблонов носит название Цикл множественной иерархии (Multiplicity Hierarchy Cycle).

Согласно данному шаблону, если между двумя классами имеется связь «наследование» (обобщение), которая означает наличие между ними отношения один-к-одному или

же несколько дочерних классов к одному родительскому классу, то и любая другая связь между данными классами также должна указывать на наличие отношения один-к-одному или же несколько дочерних классов к одному родительскому классу.

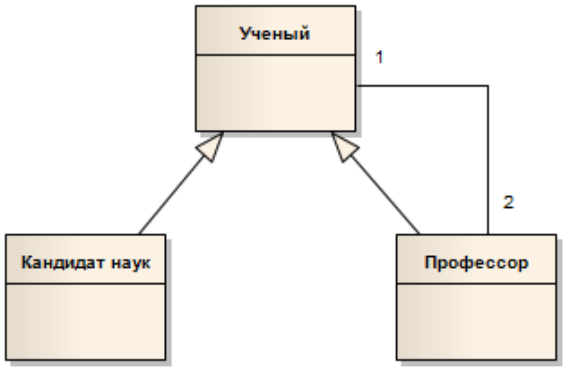

Рис. 1. Пример шаблона: Цикл множественной иерархии

На рис. 1 представлен пример данного шаблона – диаграмма классов, на которой имеется класс Ученый и классы Кандидат наук и Профессор, которые наследуются от данного класса. Таким образом, класс Кандидат наук связан с классом Ученый отношением один-к-одному или же несколько объектов класса Кандидат наук к одному классу объекта Ученый, точно так же класс Профессор связан с классом Ученый отношением один-к-одному или же несколько объектов класса Профессор к одному классу объекта Ученый. Но при этом класс Ученый связан с классом Профессор ассоциативной связью в соотношении: два объекта класса Ученый к одному классу объекта Профессор. Таким образом, между двумя классами имеются две различные связи, одна из которых указывает на наличие отношения: два объекта класса Ученый к одному классу объекта Профессор, а вторая на наличие отношения: один-к-одному или же несколько объектов класса Профессор к одному классу объекта Ученый, что противоречит друг другу и шаблону Цикл множественной иерархии.

Подразновидностью шаблона Цикл множественной иерархии является шаблон Взаимодействие циклов множественной иерархии (Interaction of multiplicity and inter-association hierarchy constraints).

Согласно данному шаблону, если между двумя классами имеется связь «наследование» (обобщение), которая означает наличие между ними отношения один-к-одному или же несколько дочерних классов к одному родительскому классу, и каждый из них ассоциативно связан с третьим классом, то данные связи должны указывать на наличие одинаковых отношений.

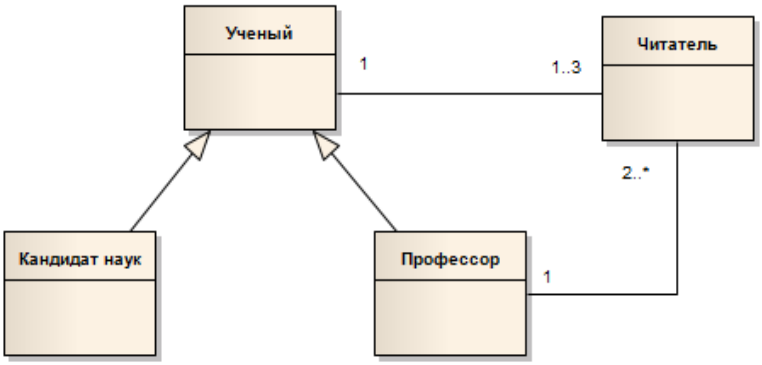

Рис. 2. Пример шаблона: Взаимодействие циклов множественной иерархии

На рис. 2 представлен пример данного шаблона – диаграмма классов, на которой представлен класс Ученый и классы Кандидат наук и Профессор, которые наследуются от данного класса, а также класс Читатель, который ассоциативно связан с классом Ученый и классом Профессор. Таким образом, класс Профессор связан с классом Ученый отношением один-к-одному, но при этом он может быть связан с количеством от 2 до бесконечности объектами класса Читатель. В то же время класс Ученый может быть связан с количеством от 1 до 3 объектов класса Читатель. Так как отношения 1 и 1 со стороны классов Ученый и Профессор удовлетворяют условию 1=1, но отношения 1...3 и 2..* со стороны класса Читатель не удовлетворяют условию (1..3)=(2..*), то связи противоречат друг другу и шаблону Простой множественный цикл.

Не менее популярный и часто встречаемый шаблон носит название Простой множественный цикл (Pure Multiplicity Cycle).

Согласно данному шаблону, если между двумя классами имеется несколько ассоциативных связей и каждая связь имеет свое отношение, то все отношения со стороны одного и того же класса должны либо быть равными, либо же включать друг друга.

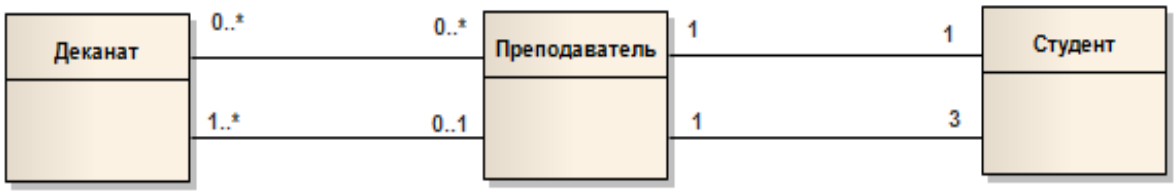

Рис. 3. Пример шаблона: Простой множественный цикл

На рис. 3 представлен пример данного шаблона – диаграмма классов, на которой представлены три класса: класс Деканат, класс Преподаватель и класс Студент. Класс Деканат связан двумя «ассоциациями» с классом Преподаватель, причем в одной «ассоциации» от 0 до бесконечности объектов класса Деканат связаны с любым числом от 0 до бесконечности объектов класса Преподаватель, а во второй – от 1 до бесконечности объектов класса Деканат связаны с 0 или 1 объектом класса Преподаватель. Так как отношения 0...* и 1..* со стороны класса Деканат удовлетворяют условию (1..*) Є (0..*), и отношения 0...* и 0..1 со стороны класса Преподаватель удовлетворяют условию (0..1) Є (0..*), то связи не противоречат друг другу и шаблону Простой множественный цикл.

В то же время класс Преподаватель связан двумя «ассоциациями» с классом Студент, причем в одной «ассоциации» 1 объект класса Преподаватель связан с 1 объектом класса Студент, а во второй – 1 объект класса Преподаватель связан с 3 объектами класса Студент. Так как отношения 1 и 1 со стороны класса Преподаватель удовлетворяют условию 1=1, но отношения 1 и 3 со стороны класса Студент не удовлетворяют ни условию 1=3, ни условию 1 Є 3, то связи противоречат друг другу и шаблону Простой множественный цикл.

4. Подход, основанный на построении идентификационного графа

Еще один подход к верификации диаграммы классов основан на построении идентификационного графа. Разработкой данного метода в период с 1990 по 2001 год занимались американские ученые Hartmann, Lenzerini, Nobili и Thalheim [5].

Суть данного метода состоит в построении идентификационного графа, представляющего собой ориентированный граф. Узлами данного графа являются классы, а также ассоциативные связи между ними, а дуги связывают ассоциации с теми классами, между которыми на диаграмме классов и указаны соответствующие ассоциативные связи. В качестве весов дуг используются отношения, которыми ассоциативно связаны классы. Как и в обычном графе, в идентификационном графе вес пути вычисляется как произведение весов всех дуг, входящих в состав данного пути.

Основным предназначением идентификационного графа является определение причины нарушения конечной выполнимости диаграммы классов. Так же, как и метод шаблонов, данный метод позволяет выявить противоречивые связи, так называемые критические

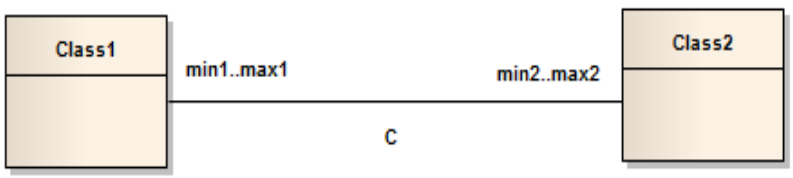

Рис. 4. Простейшая диаграмма классов

циклы.

Суть данного метода представлена на рис. 4 и 5.

На рис. 4 представлена простейшая диаграмма классов с двумя классами: Class 1 и Class 2, между которыми имеется ассоциативная связь С в отношении: от min1 до max1 объектов класса Class 1 связаны с количеством от min2 до max2 объектов класса Class 2. Идентификационный граф для этой простейшей диаграммы классов, построенный согласно данному методу, представлен на рис. 5. Граф имеет три узла: Class 1, Class 2 и С, а также четыре дуги, которые их связывают. Вес дуг слева направо и сверху вниз составляет соответственно max2, 1/min1, max1, 1/min2. Согласно данному методу, критическим, указывающим на нарушение конечной выполнимости, является цикл, вес которого меньше 1. Таким образом, если

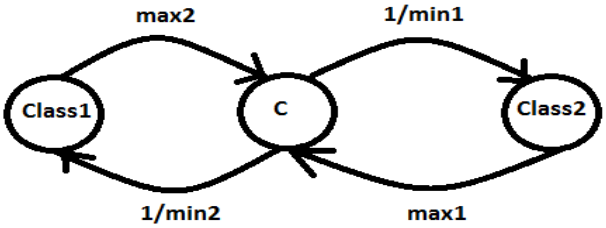

Рис. 5. Идентификационный граф для диаграммы классов с рис. 4

$$(\max 2 * 1/\min 1 * \max 1 * 1/\min 2) < 1, \quad (1)$$

то «ассоциация» содержит нарушение конечной выполнимости.

На рис. 6 представлен пример использования метода идентификационного графа для определения причины нарушения конечной выполнимости диаграммы классов. На данной диаграмме представлены два класса: класс Преподаватель и класс Студент, причем от 1 до 5 объектов класса Преподаватель «ассоциацией» Информация об успешности связаны с количеством от 1 до 30 объектов класса Студент. Идентификационный граф для данной диаграммы классов представлен на рис. 7.

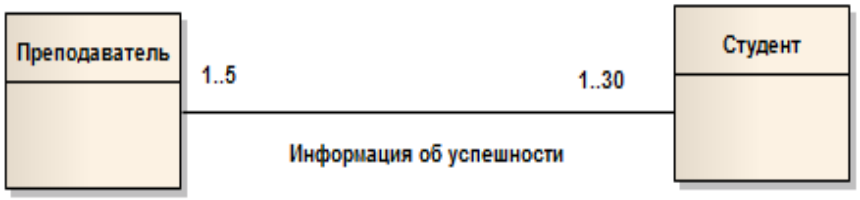

Рис. 6. Пример использования метода идентификационного графа

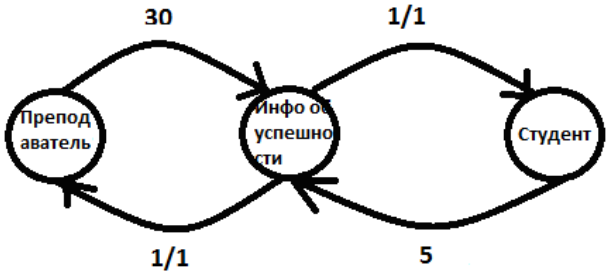

Рис. 7. Идентификационный граф для диаграммы классов с рис. 6

Тогда, согласно данному методу, критический цикл для этого идентификационного графа выглядит следующим образом:

$$(30 * 1/1 * 5 * 1/1) > 1. \quad (2)$$

Поэтому «ассоциация» не содержит нарушения конечной выполнимости.

5. Метод, основанный на представлении классов в качестве множеств

Еще один подход к верификации диаграммы классов основан на представлении класса в качестве множества. Данный метод был предложен Calvanese и Lenzerini [6] и позволяет оценить корректность прежде всего иерархий классов.

Согласно данному методу, каждый класс, входящий в состав иерархии, представляется в виде множества. Далее создается система неравенств, в которую включаются все возможные неравенства и равенства, если такие имеются, между классами-множествами. Затем необходимо решить систему неравенств и, если результатом станет не пустое подмножество, то данная иерархия классов построена корректно, если же пустое множество, иерархия классов не корректна.

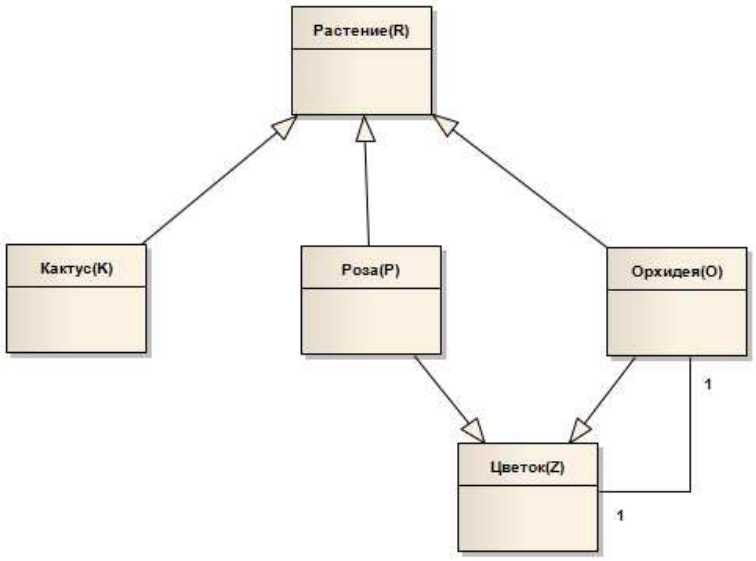

Рис. 8. Диаграмма классов, представляющая собой иерархию классов

На рис. 8 представлена диаграмма классов, которая является иерархией классов. В этой иерархии классы Роза, Орхидея и Кактус наследуются от класса Растение, в то же время классы Роза и Орхидея наследуются от класса Цветок, а класс Цветок ассоциативно связан с классом Орхидея отношением один-к-одному. Таким образом, к данной диаграмме может быть применен метод множеств. Система неравенств для данной иерархии выглядит следующим образом:

$$\begin{cases} K \cap P \cap O = 0, \\ P \cup O \subseteq Z, \\ O \subset Z, \\ |O| = |Z|. \end{cases} \tag{3}$$

Последние два неравенства в данной системе противоречат друг другу, поэтому решением данной системы неравенств является пустое множество и, согласно данному методу, иерархия классов не корректна.

6. Выводы

В статье описано несколько наиболее часто используемых в верификации диаграмм классов. Все они позволяют оценить корректность диаграммы классов с разной точки зрения. Так, метод, основанный на построении драйвера, позволяет выявить только наиболее характерные ошибки на диаграмме. Метод шаблонов позволяет оценить корректность ассоциативных связей и связей типа «обобщение». Метод, связанный с построением идентификационного графа, позволяет оценить корректность исключительно ассоциативных связей. Метод, основанный на представлении классов в качестве множеств, позволяет оценить корректность исключительно иерархий классов. Таким образом, наиболее эффективным является комплексное использование данных методов.

СПИСОК ЛИТЕРАТУРЫ

1. OMG. Architecture Board ORMSC // Model Driven Architecture (MDA). – ormsc/2001-07-01, 2001. – July 9. – 28 p.

2. Буч Г. UML. Классика CS / Буч Г., Якобсон А., Рамбо Дж.; пер. с англ.; под общей ред. проф. С. Орлова. – СПб.: Питер, 2006. – [2-е изд.]. – 736 с.
3. Макгрегор Дж. Тестирование объектно-ориентированного программного обеспечения: практ. пособ. / Дж. Макгрегор, Д. Сайкс; пер. с англ. – К.: ООО «ТИД «ДС»», 2002. – 432 с.
4. Balaban M. Management of Correctness Problems in UML Class Diagrams – Towards a Pattern-based Approach / M. Balaban, A. Maraee, A. Stur // Beer Sheva: Department of Computer Science, Ben-Gurion University of the Negev. – 2002. – 33 p.
5. Fundamentals of Entity-Relationship Modeling / S. Hartmann, M. Lenzerini, P. Nobili [et al.] // Annals Mathematics and Artificial Intelligence. – 1993. – N 7. – P. 197 – 256.
6. Calvanese D. On the Interaction between ISA and Cardinality Constraints / D. Calvanese, M. Lenzerini // Proc. of the 10th IEEE International Conference on Data Engineering. – Houston, Texas, USA. IEEE Computer Society. – Washington, DC, USA, 1994. – P. 204 – 213.

Стаття надійшла до редакції 26.03.2013