РОЗДІЛ IV. ІНФОРМАЦІЙНО-КОМП'ЮТЕРНІ ТЕХНОЛОГІЇ

УДК 004.054
DOI: 10.25140/2411-5363-2018-1(1)-68-78

Ирина Богдан, Артем Задорожний

КЛАССИФИКАЦИЯ ОШИБОК НА UML-ДИАГРАММАХ, ВОЗНИКАЮЩИХ В ХОДЕ РАЗРАБОТКИ IT-ПРОЕКТОВ

Актуальность темы исследования. *Одной из самых популярных парадигм при создании программного обеспечения является объектно-ориентированная. Создание качественного объектно-ориентированного программного обеспечения начинается с создания его модели, представленной в виде множества UML-диаграмм, и дальнейшей верификации данной модели.*

Постановка проблемы. *Существует множество различных методов верификации: одни позволяют находить отдельные группы ошибок, другие же выполняют верификацию модели в целом. Однако, для эффективного исследования этих методов необходимо прежде всего определить, какие ошибки позволяют найти те или иные методы. Наличие классификации ошибок, которые могут присутствовать на диаграммах, существенно ускорит процесс идентификации ошибок.*

Анализ последних исследований и публикаций. *Для дальнейшего использования классификации при создании и анализе методов верификации моделей программного обеспечения, классифицировать ошибки лучше всего в зависимости от их сути. На данный момент не существует единой классификации ошибок на всех базовых UML-диаграммах.*

Выделение неисследованных частей общей проблемы. *Таким образом, актуальной является задача создания классификации ошибок на UML-диаграммах, которая основана на анализе сути ошибки.*

Постановка задачи. *Главной целью данной статьи является описание классификации, основанной на анализе сути ошибок, которая бы позволила классифицировать ошибки на базовых диаграммах, к которым принадлежит диаграмма вариантов использования, диаграмма классов, диаграмма последовательности, диаграмма деятельности, диаграмма кооперации, диаграмма состояний и диаграмма компонентов.*

Изложение основного материала. *Представленная классификация ошибок на семи базовых UML-диаграммах дает возможность существенно ускорить процесс идентификации ошибок и, как результат, повысить качество создаваемого объектно-ориентированного программного обеспечения еще на этапе его проектирования.*

Выводы в соответствии со статьей. *В статье предложена классификация ошибок на UML-диаграммах, которая позволяет эффективно исследовать, а также оценивать достоинства и недостатки существующих методов верификации моделей объектно-ориентированного программного обеспечения, тем самым повышая качество создаваемой программы.*

Ключевые слова: *классификация ошибок на UML-диаграммах; верификация моделей программ; качество программного обеспечения.*

Рис.: 12. Библ.: 12.

Актуальность темы исследования. На данный момент объектно-ориентированный подход к созданию программного обеспечения является одним из наиболее распространенных. Несомненным преимуществом данного подхода является концептуальная близость к предметной области произвольной структуры и назначения. При этом не менее важной является задача обеспечения качества создаваемого программного обеспечения, которая решается уже на начальных этапах создания программы путем верификации модели будущего программного обеспечения. Данная модель при объектно-ориентированном подходе чаще всего представляется в виде UML-диаграмм [1]. Таким образом, именно с верификации UML-диаграмм и начинается создание корректно работающего и надежного программного обеспечения.

Среди UML-диаграмм, которые чаще всего принадлежат к модели программного обеспечения, диаграмма вариантов использования, диаграмма классов, диаграмма последовательности, диаграмма кооперации, диаграмма состояний, диаграмма деятельности, диаграмма компонентов и диаграмма развертывания. При этом суть процесса верификации сводится к проверке соответствия созданной диаграммы стандарту UML, а также тем требованиям, которые накладываются на нее в зависимости от используемого языка программирования, конкретной предметной области, а также решаемой задачи.

Проведенные исследования показали, что актуальным является создание классификации ошибок, которые могут встречаться на UML-диаграммах. Причем для дальнейшего ее

эффективного использования при создании и анализе методов верификации моделей программного обеспечения классифицировать ошибки лучше всего в зависимости от их сути.

Постановка проблемы. Для эффективного выполнения верификации моделей объектно-ориентированного программного обеспечения, представленных в виде UML-диаграмм, необходимо наличие классификации ошибок, которые могут присутствовать на них. Классификацию следует выполнять по такому критерию, как суть ошибки.

Анализ последних исследований и публикаций. На сегодняшний день существует множество различных методов верификации UML-диаграмм. Большинство из них выполняют верификацию отдельных диаграмм, входящих в состав модели. Так, в частности, в работе [2] описан метод шаблонов, суть которого состоит в поиске на диаграмме классов определенных конструкций – ошибок. Описанный в работе [3] метод верификации диаграммы классов основан на представлении классов в виде множеств. Метод идентификационного графа и метод антипаттернов, описанные в работе [4], также используются для верификации диаграммы классов.

В работе [5] предложены методы верификации диаграммы последовательности, такие как метод протоколов, основанный на анализе соответствующих объектам классов из диаграммы классов, и метод, основанный на построении абстрактного цифрового автомата. В работе [6] предложен метод верификации диаграммы последовательности, который основан на анализе сообщений, передаваемых от объекта к объекту.

Однако в описании ни одного из представленных методов не представлена информация о том, какие именно группы ошибок они позволяют идентифицировать.

В работе [7] предложены методы верификации моделей объектно-ориентированного программного обеспечения в целом. Описанный в [8] подход предполагает повышение качества создаваемого программного обеспечения путем выполнения верификации с использованием антипаттернов. Каждый из методов имеет свои достоинства и недостатки, однако для исследования этих методов, а также их особенностей необходимо прежде всего определить какие ошибки позволяют найти те или иные методы. Однако, для этого необходимо наличие классификации ошибок, которые могут присутствовать на той или иной диаграмме. На данный момент не существует классификации ошибок на всех UML-диаграммах.

Выделение неисследованных частей общей проблемы. Таким образом, актуальной является задача создания классификации ошибок на UML-диаграммах, которая основана на анализе сути ошибки.

Постановка задачи. Входными данными для решения задачи классификации ошибок на базовых UML-диаграммах является модель создаваемого в процессе разработки IT-проектов программного обеспечения, которая представлена в виде множества UML-диаграмм: диаграммы вариантов использования, диаграммы классов, диаграммы состояний, диаграммы деятельности, диаграммы последовательности, диаграммы кооперации и диаграммы компонентов.

Целью данного исследования является анализ и классификация ошибок на UML-диаграммах, возникающих в ходе выполнения IT-проектов, что может в дальнейшем быть использовано при создании и анализе методов верификации моделей объектно-ориентированного программного обеспечения.

Для достижения поставленной цели необходимо решить такие задачи:

1. Провести анализ ошибок на UML-диаграммах, составляющих модель разрабатываемого программного обеспечения.
2. Предложить классификацию ошибок на UML-диаграммах, основанную на анализе сути ошибки.
3. Определить пути дальнейшего исследования для детализации и расширения предложенной классификации.

Изложение основного материала. Диаграмма вариантов использования является основой для создания исходной документации и дальнейшего взаимодействия разработчиков с заказчиками или пользователями [9]. Поэтому особенно важно все возникающие на данном этапе ошибки находить и сразу устранять.

Поскольку каждый вариант использования определяет последовательность действий, которые должны быть выполнены проектируемой системой при ее взаимодействии с соответствующим актером, то каждый вариант использования обязательно должен быть описан в форме глагола. Таким образом, первая группа ошибок, которые могут быть допущены на диаграмме вариантов использования, это ошибки в описании вариантов использования. Пример данной ошибки представлен на рис. 1 – вариант использования «Контакты» описан в виде существительного, а не глагола – действия, что недопустимо.

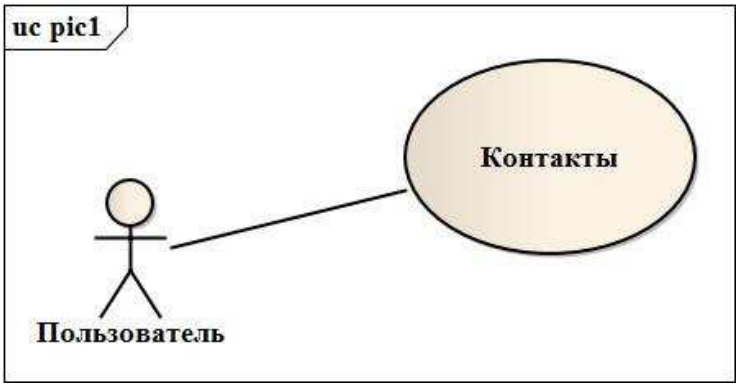

Рис. 1. Пример ошибки в описании вариантов использования

Отношения на диаграмме вариантов использования показывают взаимодействия актеров с вариантами использования системы или же поясняют отдельный вариант использования, связывая его с другими. То есть, каждый вариант использования должен относится как минимум к одному актеру или другому варианту использования. На диаграмме не могут присутствовать изолированные варианты использования, которые не связаны отношениями либо с другими вариантами использования, либо с актерами. Таким образом, вторая группа ошибок, которые могут быть допущены на диаграмме вариантов использования, это ошибки в описании отношений. Пример данной ошибки представлен на рис. 2 – вариант использования «Ввести фамилию» не связан отношением ни с каким другим вариантом использования, ни с каким-либо актером.

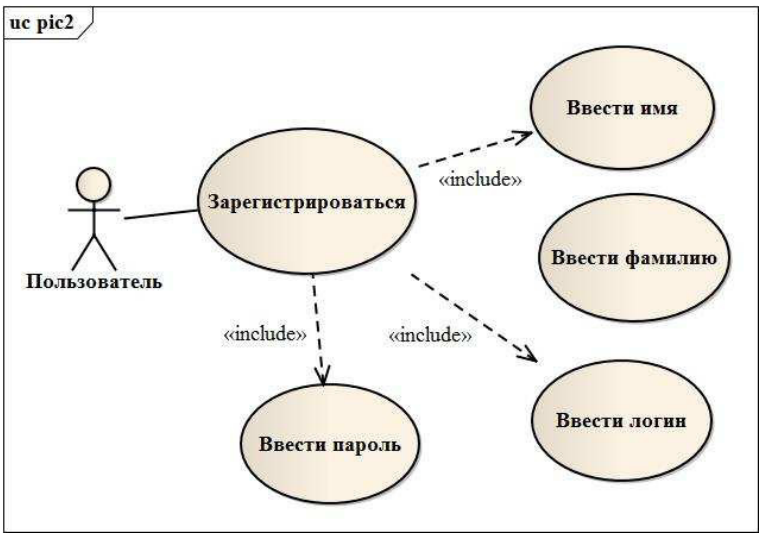

Рис. 2. Пример ошибки в описании отношений на диаграмме вариантов использования

Поскольку случаи использования, отображенные на диаграмме вариантов использования, не являются представлениями программного обеспечения и чаще всего формулируются на естественном языке, то выделить другие группы ошибок на данной диаграмме сложно.

На диаграмме классов каждый класс предназначен для описания множества объектов, которые обладают одинаковыми структурой, поведением и отношениями с объектами других классов. Каждый класс должен иметь свое уникальное в пределах одного пакета имя. Таким образом, первая группа ошибок, которые могут присутствовать на диаграмме классов, ошибки в описании имен классов. Пример данной ошибки представлен на рис. 3 – класс с именем «Order» встречается на диаграмме два раза, что не допустимо.

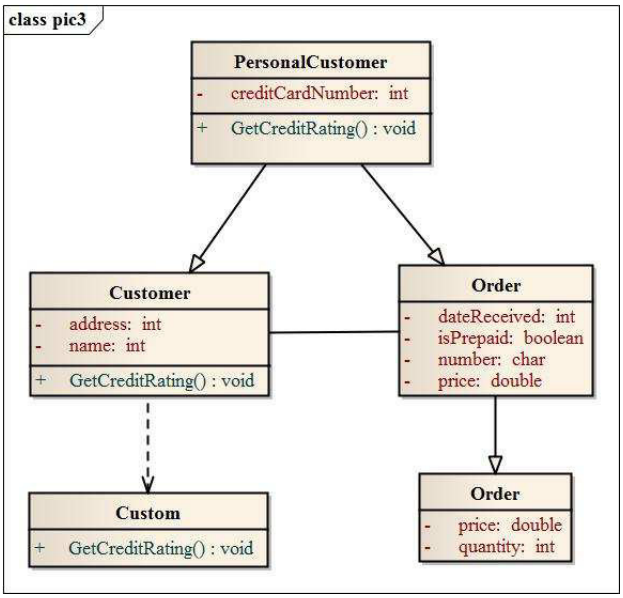

Рис. 3. Пример ошибки в описании имен классов, ошибки в описании интерфейсов и абстрактных классов, ошибки наследования на диаграмме классов

Могут присутствовать на диаграмме классов и абстрактные классы, причем они должны помечаться служебным словом «Abstract». Таким образом, следующая группа ошибок на данной диаграмме – ошибки в описании абстрактных классов. Пример данной ошибки представлен на рис. 3. Класс «Customer» по сути является абстрактным, однако не помечен служебным словом «Abstract». Таким образом, дальнейшем класс «Customer» может быть использован как класс, имеющий экземпляры, что не допустимо.

Также компонентами диаграммы классов могут являться интерфейсы, причем они должны помечаться служебным словом «Interface». Таким образом, следующая группа ошибок на данной диаграмме – ошибки в описании интерфейсов. Пример данной ошибки представлен на рис. 3. Класс «Custom» по сути является интерфейсом, однако не помечен соответствующим служебным словом. В дальнейшем «Custom» может быть использован как класс, что не допустимо.

Присутствуют на диаграмме классов и отношения. Так, отношение обобщения – это отношение между более общим элементом – родителем и более частным элементом – потомком. Однако, в некоторых объектно-ориентированных языках допустимо множественное наследование, в других – нет. Таким образом, следующая группа ошибок на данной диаграмме – ошибки наследования. Пример данной ошибки также представлен на рис. 3. Класс «Personal Customer» наследуется одновременно и от класса «Customer», и от класса «Order». Это не допустимо, если в качестве языка программирования был выбран язык Java.

Вместе с отношением ассоциации, которое показывает наличие семантической связи между классами, на диаграмме классов может отображаться и его кратность. Еще одна группа ошибок на диаграмме классов – ошибки в описании кратности. Пример данной ошибки представлен на рис. 4. Класс «Customer» связан с классом «Order» ассоциативной связью. Мощность ассоциативной связи – от 0 до 4 объектов класса «Customer» к от 6 до 2 объектов класса «Order». В данном случае, 6 меньше 2. Таким образом, кратность описана неверно.

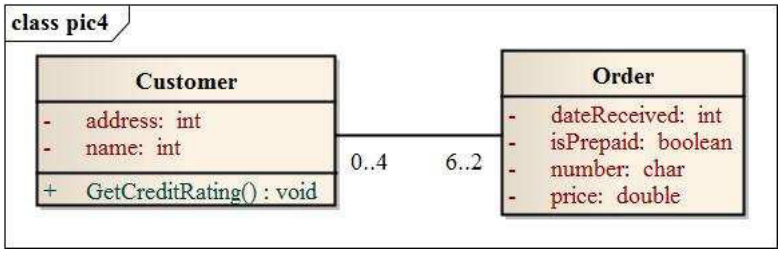

Рис. 4. Пример ошибки в описании кратности на диаграмме классов

Кроме того, на диаграмме классов допустимо наличие нескольких различных отношений между двумя классами. Еще одна группа ошибок на диаграмме классов – ошибки описания отношений. Пример данной ошибки представлен на рис. 5. Два класса «Customer» и «Order» связаны ассоциативной связью и отношением обобщения. Мощность ассоциативной связи – 4 объекта класса «Customer» к 1 объекту класса «Order». С другой стороны, класс «Order» наследуется от класса «Customer». Это означает, что классы находятся в отношении один объект класса «Customer» к одному или нескольким объектам класса «Order». То есть, отношения ассоциативной связи и обобщения противоречат друг другу.

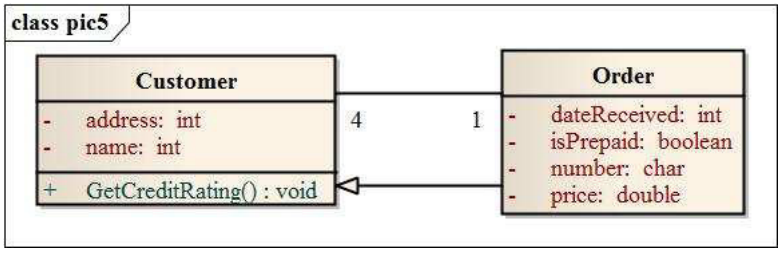

Рис. 5. Пример ошибки в описании кратности на диаграмме классов

Описанные группы ошибок описывают практически все ошибки, которые могут встретится на диаграмме классов.

На диаграмме последовательности одним из основных элементов, помимо объектов, является сообщение. Сообщения представляют собой законченный фрагмент информации, который отправляется одним объектом другому, делая тем самым объект-получатель активным.

На диаграмме последовательности могут присутствовать как синхронные, так и асинхронные сообщения. В случае отправки синхронного сообщения объект-отправитель не может выполнять каких-либо действий до получения ответа, в случае же отправки асинхронного сообщения объект-отправитель может не дожидаться ответа и продолжать свою работу. Таким образом, первая группа ошибок, которые могут присутствовать на диаграмме последовательности, это ошибки отправки синхронных сообщений. Пример данной ошибки представлен на рис. 6 – объект «Object1» отправил синхронное сообщение объекту «Object2», однако не дожидаясь от него ответа, отправил ему еще одно сообщение, что не допустимо.

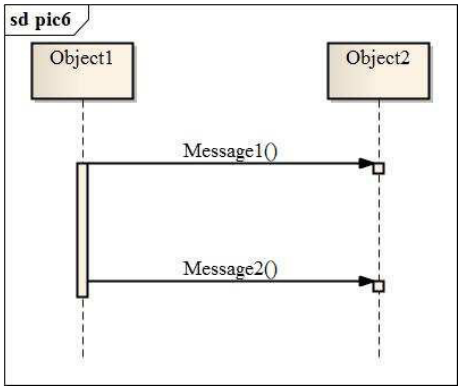

Рис. 6. Пример ошибки отправки синхронных сообщений на диаграмме последовательности

Некоторые объекты на диаграмме последовательности могут существовать на протяжении всего времени существования системы, другие создаются и уничтожаются по необходимости. При этом может возникнуть ситуация, когда сообщения будут отправляться еще не созданным или уже уничтоженным объектам. Таким образом, следующая группа ошибок на данной диаграмме – ошибки отправки сообщений не существующим объектам. Пример данной ошибки представлен на рис. 7 – объект «Object1» отправил сообщение объекту «Object2», однако на момент отправки последний уже был удален.

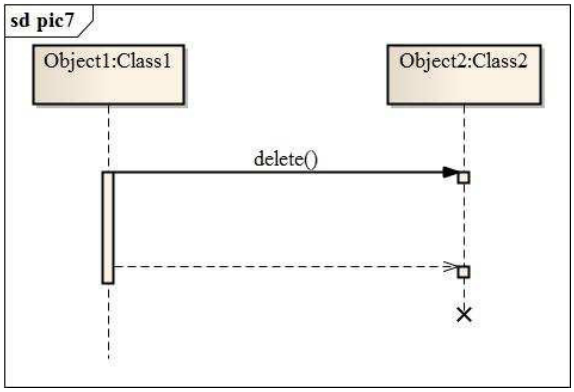

Рис. 7. Пример ошибки отправки сообщений не существующим объектам на диаграмме последовательности

Все сообщения, которые представлены на диаграмме последовательности, могут быть логически выведены из пред- и постусловий, регламентирующих выполнение отдельных операций, объявленных в классе этого объекта и представленных на соответствующей диаграмме классов. Еще одной группой ошибок на диаграмме последовательности являются ошибки отправки не существующих сообщений. На рис. 8 представлен пример простейшей диаграммы классов, состоящей из двух классов, каждый из которых имеет по два метода. На рис. 9 представлена диаграмма последовательности, на которой объект «Object2» получает от объекта «Object1» сообщение «m1», однако на соответствующей диаграмме классов (рис. 8) нет метода, который бы соответствовал данному сообщению.

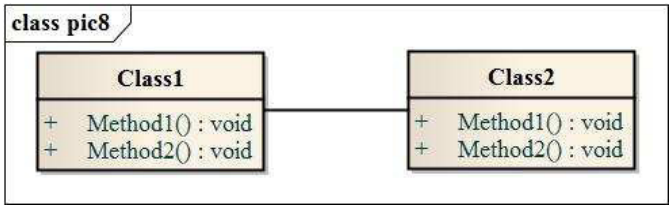

Рис. 8. Простейшая диаграмма классов

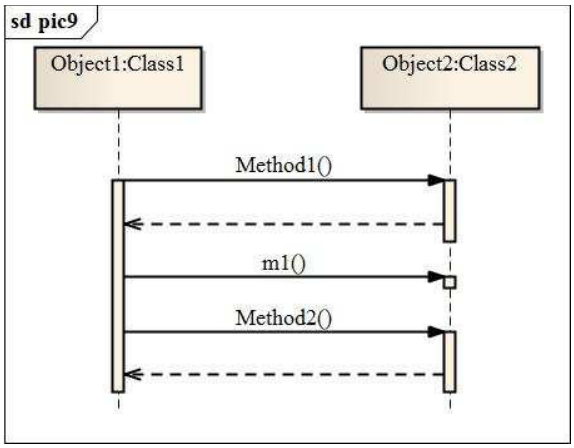

Рис. 9. Пример ошибки отправки не существующих сообщений на диаграмме последовательности

Помимо простых сообщений на диаграмме последовательности могут присутствовать также и ветвления, при этом необходимо контролировать, чтоб условия переходов в ветвлении не противоречили друг другу. Таким образом, следующей группой ошибок являются ошибки ветвления. Пример данной ошибки представлен на рис. 10 – объект «Object1» отправил сообщение-ветвление, которое в случае выполнения соответствующего условия может передать управление либо объекту «Object2», либо объекту «Object3». Однако в том случае, если, к примеру, x=8 оба условия выполняются и оба объекта становятся активными одновременно, что недопустимо.

В то время, как диаграмма последовательности служит для визуализации временных аспектов взаимодействия, диаграмма кооперации предназначена для спецификации структурных аспектов взаимодействия без учета фактора времени [10]. То есть, на диаграмме кооперации изображаются только отношения между объектами, играющими определенные роли во взаимодействии. На этой диаграмме не указывается время в виде отдельного измерения. Таким образом, все группы ошибок, которые могут присутствовать на диаграмме последовательности, характерны также и для диаграммы кооперации.

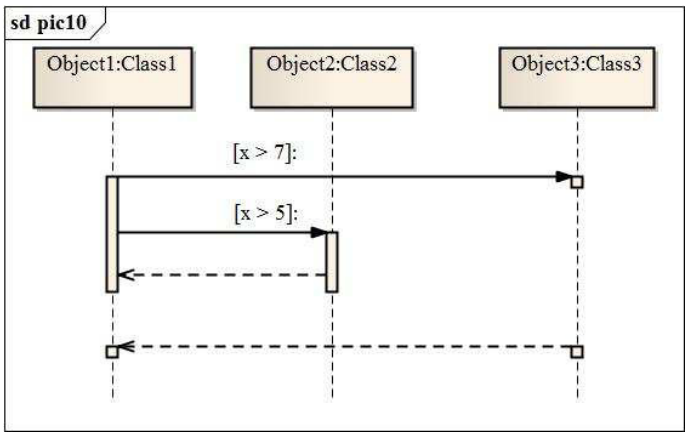

Рис. 10. Пример ошибки ветвления на диаграмме последовательности

Описанные группы ошибок описывают практически все ошибки, которые могут встретится на диаграммах последовательности и кооперации.

Диаграмма состояний представляет собой граф специального вида – цифровой автомат. Соответственно, и требования к диаграмме состояний соответствуют требованиям к автомату.

Количество состояний на диаграмме состояний должно быть конечным и все состояния должны быть описаны явным образом. Следовательно, первая группа ошибок, которые могут встречаться на диаграмме состояний – это ошибки в описании состояний.

Также диаграмма не может содержать изолированных состояний и конфликтующих переходов, то есть если из одного состояния возможен переход в одно из нескольких других, то такой переход должен быть триггерным – условным. Таким образом, вторая группа ошибок на диаграмме состояний – ошибки переходов из состояния в состояние. Пример данной ошибки представлен на рис. 11. Состояние «Invalid» является изолированным, в него нельзя перейти не из какого другого, что недопустимо. С другой стороны, из состояния «DialTone» можно перейти сразу в два состояния «TimeOut» и «Dialing», что также недопустимо.

Применяемая на диаграмме деятельности графическая нотация во многом похожа на нотацию диаграммы состояний, поскольку на диаграмме деятельности также присутствуют обозначения состояний и переходов. Отличие заключается в семантике состояний, которые используются для представления не деятельности, а действий, и в отсутствии на переходах сигнатуры событий [11]. Каждое состояние на диаграмме деятельности соответствует выполнению некоторой элементарной операции, а переход в следующее состояние срабатывает только при завершении этой операции в предыдущем состоянии. Таким образом, все группы ошибок, которые могут присутствовать на диаграмме состояний, характерны также и для диаграммы деятельности.

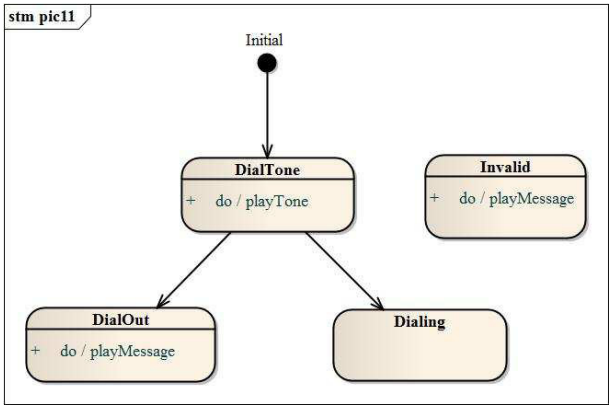

Рис. 11. Пример ошибки переходов из состояния в состояние на диаграмме состояний

Основными элементами на диаграмме компонентов являются компоненты трех различных видов, при этом:

– компоненты со стереотипом «table» не могут быть непосредственно связаны с какими-либо компонентами, только через компонент со стереотипом «DB»;

– компоненты-рабочие продукты не могут быть непосредственно связаны с компонентами со стереотипом «DB», только через компоненты исполнения.

Таким образом, на диаграмме компонентов могут присутствовать ошибки, принадлежащие к одной группе – ошибки связи между компонентами. Пример данной ошибки представлен на рис. 12 – компонент «Order» со стереотипом «table» связан зависимостью с компонентом-рабочим продуктом «Order», что недопустимо.

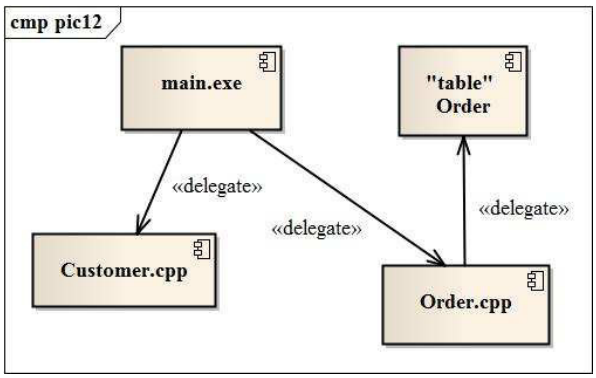

Рис. 12. Пример ошибки связи между компонентами на диаграмме компонентов

Описанные группы ошибок описывают практически все ошибки, которые могут встретится на диаграммах состояний и деятельности.

Диаграмма развертывания является самой «демократичной» из всех UML-диаграмм. Помимо канонического представления компонентов на ней разрешается использование и графических примитивов. Отсюда следует, что выделить на данной диаграмме какие-либо группы ошибок крайне сложно.

Выводы в соответствии со статьей. В статье решена задача классификации ошибок на восьми базовых UML-диаграммах. Данная классификация позволяет отнести все ошибки, выявленные на диаграмме вариантов использования, диаграмме классов, диаграмме последовательности, диаграмме кооперации, диаграмме состояний, диаграмме деятельности и диаграмме компонентов к той или иной группе.

Научная новизна работы состоит в том, что впервые предложена классификация ошибок на UML-диаграммах, которая основана на анализе сути ошибок, что позволило использовать ее при создании и анализе методов верификации моделей объектно-ориентированного программного обеспечения.

Практическая ценность полученных результатов состоит в том, что использование описанной классификации в процессе анализа UML-диаграмм позволит существенно ускорить процесс поиска ошибок, и, как результат, повысить качество создаваемого программного обеспечения еще на этапе его проектирования.

Дальнейшее исследование в данном направлении может быть направлено на расширение данной классификации либо путем добавления новых групп ошибок на базовых UML-диаграммах, либо путем классификации ошибок на других диаграммах стандарта UML.

Список использованных источников

1. *Page-Jones M.* Fundamentals of Object-Oriented Design in UML / M. Page-Jones. – New York : Addison-Wesley, 2000. – 435 p.

2. *Balaban M.* Management of Correctness Problems in UML Class Diagrams – Towards a Pattern-based Approach / M. Balaban, A. Maraee, A. Sturm // Journal of Information System Modeling and Design. – 2010. – Vol. 1. – No 4. – P. 24–47.

3. *Maraee, A.* Efficient recognition of finite satisfiability in UML class diagram: strengthening by propagation of disjoint constraints / A. Maraee, M. Balaban // Proceedings International Conference Model-Based Systems Engineering MBSE. – 2009. – P. 1–8.

4. *Литвинов В. В.* Формальная верификация диаграммы классов / В. В. Литвинов, И. В. Богдан // Математические машины и системы. – 2013. – № 2. – С. 41–47.

5. *Lytvynov V.* Formal verification of the sequence diagram / Vitaliy Lytvynov, Irina Bogdan // International journal “Information content and processing”. – 2014. – № 1. – P. 79–86.

6. *Lima V.* Formal verification and validation of UML 2.0 sequence diagrams using source and destination of messages / V. Lima, C. Talhi, D. Mouheb, M. Debbabi, L. Wang // Electronic notes in theoretical computer science. – 2009. – № 254. – P. 143–160.

7. *Lopes L.* Object oriented modeling of multistage stochastic linear programs / L. Lopes, R. Fourer // Operations research and cyber-infrastructure. – Springer Science+Business Media, LLC, 2009. – P. 21–42.

8. *El-Attar M.* Improving the quality of the use case models using antipatterns / M. El-Attar, J. Miller // Software systems modeling. – 2010. – Vol. 9, Issue 2. – P. 141–160.

9. *Макгрегор Дж.* Тестирование объектно-ориентированного программного обеспечения. Практическое пособие / Дж. Макгрегор, Д. Сайкс ; пер. с англ. – К. : ООО «ТИД “ДС”», 2002. – 432 с.

10. *Schafer T.* Model checking UML State Machines and collaborations / T. Schafer, A. Knapp, S. Merz // Electronic Notes in Theoretical Computer Science. – 2001. – No 47. – P. 1–13.

11. *Storrle H.* Semantics and Verification of Data Flow in UML 2.0 Activities / Harald Storrle // Electronic Notes in Theoretical Computer Science. – 2005. – Vol. 127. – No 4. – P. 35–52.

12. *Литвинов В. В.* Автоматизированная система верификации моделей объектно-ориентированного программного обеспечения / В. В. Литвинов, И. В. Богдан // Вісник Чернігівського державного технологічного університету. Серія «Технічні науки». – 2015. – № 1 (77). – С. 83–90.

References

1. Page-Jones, M. (2000). *Fundamentals of Object-Oriented Design in UM*. New York: Addison-Wesley.

2. Balaban, M., Maraee, A., Sturm, A. (2010). Management of Correctness Problems in UML Class Diagrams – Towards a Pattern-based Approach. *Journal of Information System Modeling and Design*, 4, 24– 47.

3. Maraee, A., Balaban, M. (2009). Efficient recognition of finite satisfiability in UML class diagram: strengthening by propagation of disjoint constraints. Proceedings *International Conference Model-Based Systems Engineering MBSE* (pp. 1–8).

4. Lytvynov, V., Bogdan, I. (2013). Formalnaia verifikatsiia diagrammy klassov [Formalnaia verifikatsiia diagrammy klassov]. *Matematicheskie mashiny i sistemy – Mathematical Machines and Systems,* 2, 41–47 [in Russian].

5. Lytvynov, V., Bogdan, I. (2014). Formal verification of the sequence diagram. *International journal "Information content and processing"*, 1, 79–86.

6. Lima, V., Talhi, C., Mouheb, D., Debbabi, M., Wang, L. (2009). Formal verification and validation of UML 2.0 sequence diagrams using source and destination of messages. *Electronic notes in theoretical computer science,* 254, 143–160.

7. Lopes, L., Fourer, R. (2009). Object oriented modeling of multistage stochastic linear programs. *Operations research and cyber-infrastructure* (pp. 21–42).

8. El-Attar, M., Miller, J. (2010) Improving the quality of the use case models using antipatterns. *Software systems modeling,* Vol. 9, Issue 2, 141–160.

9. Makgregor, Dzh., Saiks, D. (2002). *Testirovanie obieektno-orientirovannogo programmnogo obespecheniia [Testirovanie obieektno-orientirovannogo programmnogo obespecheniia]*. Kyiv: OOO "TID «DS»".

10. Schafer, T., Knapp, A., Merz, S. (2001). Model checking UML State Machines and collaborations. *Electronic Notes in Theoretical Computer Science*, 47, 1–13.

11. Storrle, H. (2005). Semantics and Verification of Data Flow in UML 2.0 Activities. *Electronic Notes in Theoretical Computer Science*, 4, 35–52.

12. Lytvynov, V., Bogdan, I. (2015). Avtomatizirovannaia sistema verifikatsii modelei obieektno-orientirovannogo programmnogo obespecheniia [Avtomatizirovannaia sistema verifikatsii modelei obieektno-orientirovannogo programmnogo obespecheniia]. *Visnik Chernihivskoho derzhavnoho tekhnolohichnoho universitetu. Seriia "Tekhnichni nauky" – Visnyk of Chernihiv State Technological University. Series "Technical Sciences"*, 1 (77), 83–90 [in Russian].

UDK 004.054

Iryna Bohdan, Artem Zadorozhnii

THE CLASSIFICATION OF ERRORS ON UML-DIAGRAMS OCCURING IN THE DEVELOPMENT OF IT-PROJECTS

Urgency of the research. *One of the most popular paradigms for creating software is object-oriented one. The creation of the qualified object-oriented software begins with the creation of its model, presented as a set of UML-diagrams and the further verification of this model.*

Target setting. *There are many different verification methods available: some allow you to find separate groups of errors, while the others perform the verification of the model as a whole. However, for an effective study of these methods, it is necessary to determine which errors allow to find these or those methods, on the first place. The existence of the classification of errors that may be found in the diagrams can significantly improve the process of the identification of the errors.*

The analysis of actual scientific researches and issues. *For the further usage of the classification while creating and analysing the methods for verifying software models, it is better to classify errors according to their nature. Nowadays there is no single classification of errors on all basic UML diagrams.*

The defining of previously unresolved parts of a general problem. *Thus, the actual task is to create a classification of errors on UML diagrams, which is based on the analysis of the nature of the error.*

The research objective. *The main purpose of this article is to describe the classification which is based on the analysis of the nature of the errors that would allow to classify errors in the basic diagrams which includes the diagram of the use options, the class diagram, the sequence diagram, the activity diagram, the cooperation diagram, the state diagram and the component diagram.*

The description of basic material. *The given classification of errors in seven basic UML diagrams allows you to accelerate the process of identifying errors significantly and, as a result, to improve the quality of the created object-oriented software at the stage of its design.*

Conclusions. *The article represents the classification of errors in UML diagrams, which allows to investigate effectively, as well as to evaluate the advantages and disadvantages of the existing methods of models verification of the object-oriented software, which means the improvement of the program quality.*

Keywords: *classification of errors in UML diagrams; verification of program models; software quality.*

Fig.: 12. References: 12.

УДК 004.054

Ірина Богдан, Артем Задорожній

КЛАСИФІКАЦІЯ ПОМИЛОК НА UML-ДІАГРАМАХ, ЩО ВИНИКАЮТЬ ПІД ЧАС РОЗРОБКИ ІТ-ПРОЕКТІВ

Актуальність теми дослідження. *Однією з найпопулярніших парадигм при створенні програмного забезпечення є об'єктно-орієнтована. Створення якісного об'єктно-орієнтованого програмного забезпечення починається зі створення його моделі, представленої у вигляді множини UML-діаграм, і подальшої верифікації даної моделі.*

Постановка проблеми. *Існує безліч різних методів верифікації: одні дозволяють знаходити окремі групи помилок, інші ж виконують верифікацію моделі загалом. Однак для ефективного дослідження цих методів необхідно насамперед визначити, які помилки дозволяють знайти ті чи інші методи. Наявність класифікації помилок, які можуть бути присутніми на діаграмах, істотно прискорить процес ідентифікації помилок.*

Аналіз останніх досліджень і публікацій. *Для подальшого використання класифікації в процесі створення та аналізу методів верифікації моделей програмного забезпечення, класифікувати помилки найкраще залежно від їх суті. Нині немає єдиної класифікації помилок на всіх базових UML-діаграмах.*

Виділення недосліджених частин загальної проблеми. *Таким чином, актуальною є задача створення класифікації помилок на UML-діаграмах, яка б базувалася на аналізі суті помилки.*

Постановка завдання. *Головною метою даної статті є опис класифікації, заснованої на аналізі суті помилок, яка б дозволила класифікувати помилки на базових діаграмах, до яких належить діаграма варіантів використання, діаграма класів, діаграма послідовності, діаграма діяльності, діаграма кооперації, діаграма станів і діаграма компонентів.*

Виклад основного матеріалу. *Представлена класифікація помилок на семи базових UML-діаграмах дає можливість істотно прискорити процес ідентифікації помилок і, як результат, підвищити якість створюваного об'єктно-орієнтованого програмного забезпечення ще на етапі його проектування.*

Висновки відповідно до статті. *У статті запропоновано класифікацію помилок на UML-діаграмах, яка дозволяє ефективно досліджувати, а також оцінювати переваги й недоліки існуючих методів верифікації моделей об'єктно-орієнтованого програмного забезпечення, тим самим підвищуючи якість створюваної програми.*

Ключові слова: *класифікація помилок на UML-діагарамах; верифікація моделей програм; якість програмного забезпечення.*

Рис.: 12. Бібл.: 12.

Богдан Ирина Валентиновна – кандидат технических наук, доцент кафедры информационных технологий и программной инженерии, Черниговский национальный технологический университет (ул. Шевченко, 95, г. Чернигов, 14035, Украина).

Богдан Ірина Валентинівна – кандидат технічних наук, доцент кафедри інформаційних технологій та програмної інженерії, Чернігівський національний технологічний університет (вул. Шевченко, 95, м. Чернігів, 14035, Україна).

Bohdan Iryna – PhD in Technical Sciences, Associate Professor of Department of Information Technologies and Software Engineering, Chernihiv National University of Technology (95 Shevchenka Str., 14035 Chernihiv, Ukraine).

E-mail: irakirienko@gmail.com

ORCID: http://orcid.org/0000-0003-1521-6958

Задорожний Артем Александрович – кандидат технических наук, доцент кафедры информационных технологий и программной инженерии, Черниговский национальный технологический университет (ул. Шевченко, 95, г. Чернигов, 14035, Украина).

Задорожній Артем Олександрович – кандидат технічних наук, доцент кафедри інформаційних технологій та програмної інженерії, Чернігівський національний технологічний університет (вул. Шевченко, 95, м. Чернігів, 14035, Україна).

Zadorozhnii Artem – PhD in Technical Sciences, Associate Professor of Department of Information Technologies and Software Engineering, Chernihiv National University of Technology (95 Shevchenka Str., 14035 Chernihiv, Ukraine).

E-mail: zaotroy@gmail.com

ORCID: http://orcid.org/0000-0002-3424-7293

Богдан И. Классификация ошибок на UML-диаграммах, возникающих в ходе разработки IT-проектов / И. Богдан, А. Задорожний // Технічні науки та технології. – 2018. – № 1 (11). – С. 68-78.