УДК 004.031.6

В.В. Литвинов, д-р техн. наук
И.В. Богдан, ассистент
К.С. Сливко, магистрант
Черниговский государственный технологический университет, г. Чернигов, Украина

ИНСТРУМЕНТАЛЬНЫЕ СРЕДСТВА ВЕРИФИКАЦИИ МОДЕЛЕЙ ПРОГРАММНОГО ОБЕСПЕЧЕНИЯ

В.В. Литвинов, д-р техн. наук
І.В. Богдан, асистент
К.С. Слівко, магістрант
Чернігівський державний технологічний університет, м. Чернігів, Україна

ІНСТРУМЕНТАЛЬНІ ЗАСОБИ ВЕРИФІКАЦІЇ МОДЕЛЕЙ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

V.V. Lytvynov, Doctor of Technical Sciences
I.V. Bohdan, assistant
K.S. Slivko, undergraduate
Chernihiv State Technological University, Chernihiv, Ukraine

SOFTWARE MODELS OF TOOLS VERIFICATION

Обоснована необходимость проверки правильности диаграмм на этапе проектирования программного обеспечения. Рассмотрены наиболее известные средства, позволяющие проводить верификацию UML-диаграмм. Проведена сравнительная оценка существующих средств. Основными критериями оценки являются открытость кода, точность проверки, количество задействованных методов верификации.

***Ключевые слова**: верификация, диаграмма, UML, открытое программное обеспечение.*

Обґрунтовано необхідність перевірки правильності діаграм на етапі проектування програмного забезпечення. Розглянуто найбільш відомі засоби, що дозволяють проводити верифікацію UML-діаграм. Проведене порівняльне оцінювання існуючих засобів. Основними критеріями оцінювання є відкритість коду, точність перевірки, кількість задіяних методів верифікації.

***Ключові слова**: верифікація, діаграма, UML, відкрите програмне забезпечення.*

The necessity of checking of the correctness of the diagrams in the design phase of the software development was justified. The most well-known instruments to allow verification of UML-diagrams were considered. A comparative evaluation of existing funds was conducted. The main criterias are the openness of the code, the accuracy of testing, the number of involved methods of verification.

***Key words**: verification, chart, UML, open source software.*

Постановка проблемы. На данный момент существует достаточно много инструментальных средств, позволяющих выполнять как различные виды тестирования, так и несколько видов тестирования программного обеспечения в комплексе.

Однако существует небольшое количество инструментальных средств, которые выполняют верификацию программного обеспечения и, в частности, верификацию моделей программного обеспечения, где под моделью чаще всего подразумевается множество UML-диаграмм (диаграмма вариантов использования, диаграмма классов, диаграмма последовательности, диаграмма состояний и другие диаграммы из базового набора и не только).

Анализ последних исследований и публикаций. Верификация моделей является относительно новой сферой исследований, поэтому соответствующих публикаций существует крайне мало. Наиболее известной книгой по данной тематике является «Тестирование объектно-ориентированного программного обеспечения» Дж. Макгрегора. В ней он предлагает основные методы верификации.

Выделение не решенных ранее частей общей проблемы. Средство верификации программного обеспечения должно позволять:

– максимально точно оценить правильность модели;

– сводить к минимуму количество ошибок на этапе разработки.

Цель статьи. Главной целью данной работы является обзор существующих инструментальных средств верификации моделей программного обеспечения, выделение осно-

вных положительных и отрицательных качеств каждого средства, а также введение нового средства, позволяющего обеспечивать максимально корректную верификацию.

Изложение основного материала. Среди существующих инструментальных средств верификации моделей программного обеспечения можно выделить такие:

ArgoUML

ArgoUML – приложение для создания UML-диаграмм, написанное на языке Java и выпущенное под лицензией открытого программного обеспечения – EclipsePublicLicense (EPL) [1]. Так как это Java-приложение, оно доступно на любой платформе, поддерживающей Java.

ArgoUML пока не полностью реализовал стандарты UML.

Функциональность ArgoUML включает в себя:

1. Все 9 базовых диаграмм UML 1.4 поддерживаются, хотя еще не реализованы. Только диаграмма классов и диаграмма вариантов использования представлены в более или менее полном объеме.
2. Платформо-независимость (начиная с Java 1.5).
3. Нет необходимости в установке, можно работать через браузер.
4. Поддержка стандарта UML 1.4.
5. Поддержка XMI.
6. Возможность экспорта диаграмм в GIF, PNG, PS, EPS, SVG и PGML.
7. Возможность работы на десяти языках: EN, EN-GB, DE, ES, IT, RU, FR, PT, NB, ZH.
8. Интерфейс является интуитивно понятным и удобным для использования.
9. Автоматическая верификация UML-модели.
10. Поддержка OCL.
11. Генерация исходного кода Java, C++, C# и PHP.
12. Обратный инжиниринг из исходного кода и байткода Java.

К недостаткам данного приложения можно отнести нереализованность части функционала, а также отсутствие поддержки UML 2.

TelelogicTAUG2

TAUG2 от Telelogic – это средство моделирования, которое сочетает в себе мощь и простоту использования, а также предоставляет возможность начальной верификации и симуляции создаваемых моделей. Однако данный продукт не является широко распространенным и популярным. Одной из основных причин этому является его проприетарность.

TAU позволяет создавать все виды диаграмм UML 2.0, проверять их корректность и синтаксическую правильность, симулировать выполнение диаграмм, экспортировать и печатать диаграммы и многое другое. Работает на таких платформах, как:

– Windows 2000 Professional,
– Windows XP,
– SunSolaris,
– Redhat Enterprise Linux,
– Citrix XPe.

Поддерживаются такие компиляторы: Microsoft Visual Studio. NET, Wind River Systems C/C++, gnu gcc, Sun Studio 8 C/C++, Java SDK**, Green Hills MULTI C, C++.

TAU интегрируется в такие распространенные среды программирования, как Microsoft Visual Studio. NET и Eclipse.

Что касается редакций пакета, то (не считая специализированных версий) их три:

1. TAU/ModelAuthor.

Это продвинутая среда моделирования UML 2.0, включающая проверку синтаксиса и семантики, что позволяет планировщикам и архитекторам создавать точные, простые для понимания и логичные спецификации.

2. TAU/Architect.

Добавлена поддержка SysML, динамической симуляции выполнения моделей и их верификации.

3. TAU/Developer.

Добавлена возможность генерации кода для C, C++ или Java, что позволит разработчикам работать более эффективно. Telelogic поддерживает UML 2.0, что уже дает ему преимущество над ArgoUML.

К недостаткам Telelogic можно отнести отсутствие каких-либо демо- или пробных версий с ограниченной функциональностью. Так как данное ПО проприетарное, что ограничивает возможность его использования и распространения.

Visual Paradigm

Visual Paradigm for UML – это профессиональный инструмент для работы с UML, который поддерживает весь рабочий цикл программы – анализ, ориентированный на объекты, ориентированный на объекты дизайн, конструкцию, тестирование и разработку [2].

С помощью программы UML моделирования можно создавать все типы классовых диаграмм, просматривать в обратном порядке код, генерировать код с диаграмм и генерировать документацию.

Важным преимуществом является сервисная программа UMLCASE, которая предоставляет удобные UML-руководства, интерактивные UML-демонстрации и UML-проекты.

Работает на таких платформах, как:

– Window,

– Linux,

– Mac OS X.

Поддерживает проверку правильности диаграмм.

Недостатком является коммерческое использование, а также необходимость дополнительно скачивать документацию.

Enterprise Architect

Одним из наиболее популярных средств для создания UML-диаграмм является Enterprise Architect.

Данную среду разработки также можно использовать для верификации UML-диаграмм по правилам UML. В Enterprise Architect верификация диаграммы включена в понятие «валидация». Правила же UML можно указать в специальном диалоговом окне Model Validation Configuration, также как и любые ограничения, при использовании языка Object Constraint (OCL). Можно обеспечить проверку одного элемента UML, диаграммы или всего пакета.

Валидация в пакете UML может осуществляться на нескольких уровнях:

– Уровень элементов. Происходит проверка элемента и его потомков характеристик (атрибуты и операции) и отношений (связи).

– Уровень диаграмм. Проверяется сама диаграмма (ее корректность), а также любые элементы и связи в диаграмме.

– Уровень пакетов. Проверяется пакет и все подпакеты, элементы, связи и диаграммы в нем.

Enterprise Architect выполняет проверку и отображает результаты в окне вывода. Он имеет удобный графический интерфейс для визуализации процесса валидации, который можно остановить в любой момент.

На рисунке 1 показан пример ошибки, которую может отследить Enterprise Architect. Класс не может наследовать сам себя.

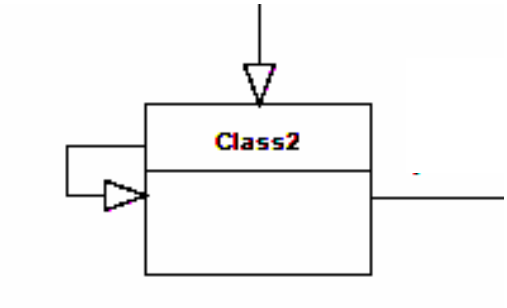

Рис. 1. Пример некорректной связи на диаграмме

Однако Enterprise Architect является проприетарным программным обеспечением, соответственно, доступен не всем, не позволяет проследить алгоритм работы и увидеть исходный код. Также существует мало документации по верификации для данного пакета, что также может вызвать некоторые затруднения при работе.

UMLTester

UMLTester – это инструментальное средство верификации, которое позволяет оценивать корректность UML-диаграмм. Оно обеспечивает верификацию диаграмм классов и диаграмм последовательности. Однако в перспективе, помимо указанных, оно будет обеспечивать проверку всех базовых диаграмм UML: диаграммы вариантов использования, диаграмма состояний и т. д.

Так как это Java-приложение, оно работает на любой платформе, которая поддерживает Java (начиная с Java 1.5). Диаграммы классов и последовательности должны поступать на вход в комплексе, поскольку верификация диаграммы последовательности выполняется при использовании результатов тестирования классов.

Так как для приложения необходима утилита, которая бы являлась открытой и имела возможность создания максимального количества существующих UML-диаграмм, используется утилита uml2 modelingtools, которая является бесплатной, обеспечивает создание базовых диаграмм, а также интегрируется с открытой средой разработки Eclipse.

Ниже представлена архитектура данного средства (рис. 2).

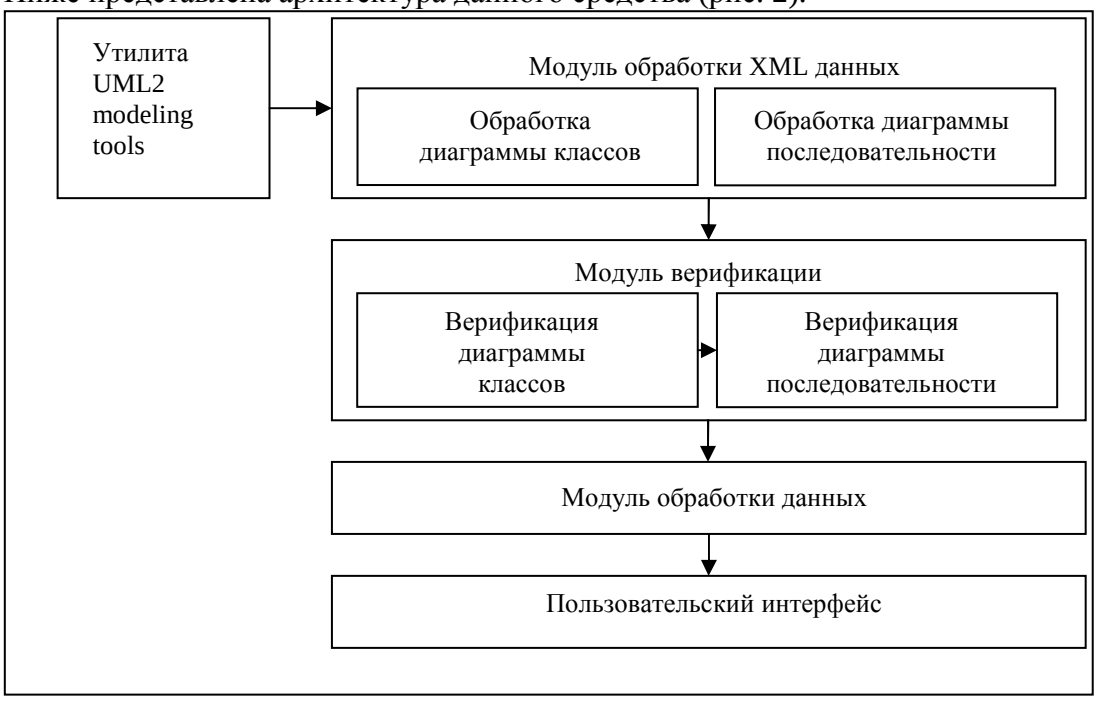

Рис. 2. Архитектура инструментального средства «UML Tester»

Ниже представлено описание модулей инструментального средства «UML Tester”:

Модуль обработки XML данных

При создании диаграммы с помощью выбранной утилиты, появляются два файла: непосредственно сама диаграмма, а также файл *.uml, в котором диаграмма представляется в виде XML-файла. Именно этот файл используется далее для парсинга диаграммы, поскольку формат XML является одним из простейших и существенно упрощает работу разработчика.

После получения нужных диаграмм в формате XML, данные XML обрабатываются и представляются в виде java объектов. Получение этих объектов происходит при помощи парсеров с использованием библиотеки dom4j, которая позволяет парсить DOM-дерево для Java.

Парсер диаграммы классов позволяет получить названия классов, их идентификаторы, названия методов и атрибутов. Что касается парсера диаграммы последовательности, он позволяет получить линии жизни, сообщения, их идентификаторы, фрагменты, показывающие процесс передачи сообщений и их последовательность.

Модуль верификации

Идея, используемая в предлагаемом средстве верификации, состоит в том, что существующие и предложенные разработчиками инструментальные средства верификации диаграмм используются в комплексе.

В настоящее время существуют такие основные подходы к определению корректности диаграмм классов [3]:

– *метод линейных неравенств.* Метод используется для диаграммы классов, которая включает типы сущностей (классы), n-арные типы отношений (ассоциации), и множественность ограничений. Он основывается на задаче нахождения решений системы линейных неравенств, для которых переменными являются количество экземпляров для типов сущностей и типов отношений;

– *метод детектирующих (идентификационных) графов*. В методе используется ориентированный граф, узлы которого соответствуют классам и ассоциациям классовой диаграммы, а его дуги соединяют ассоциативные узлы с соответствующими им классовыми узлами. Цикл графа, вес которого меньше 1, указывает на невыполнимый набор ограничений;

– для объяснения причин некорректности и корректировки диаграмм классов также используется *метод шаблонов*. Правильнее сказать, что данный метод основывается на *антишаблонах* – плохих решениях типичных проблем. Они указывают на отрицательные проектные решения и предлагают различные способы избавления от этих решений;

– *метод множеств.* 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.

Что касается диаграмм последовательности, то существует три основных метода их верификации:

– *метод протоколов.* Идея данного метода заключается в том, что сообщения, находящиеся на диаграмме классов, сопоставляются с методами из диаграммы классов, после чего осуществляется проверка, соответствует данное сообщение какому-либо методу диаграммы классов или нет. В случае отсутствия соответствия, верификация не проходит;

– *метод создания тестового драйвера.* Суть метода состоит в том, что если объекту приходит какое-либо сообщение до сообщения о создании или после сообщения об удалении, то верификация не проходит. Также верификация не проходит в случае, если синхронное сообщение не получает последующего ответа;

– *метод сценариев*. Идея метода состоит в том, чтобы представить диаграмму в виде конечного автомата и тестировать его согласно определению и его свойствам [4].

Модуль обработки данных

В данном модуле происходит непосредственная идентификация ошибок, а также получение результата о правильности диаграммы в целом.

Пользовательский интерфейс

Тут происходит оптимизация результатов и реализация удобного пользовательского интерфейса.

Выводы и предложения. Рассмотрев указанные продукты, позволяющие осуществлять верификацию диаграмм, можно утверждать, что TAUG2 от Telelogic, Enterprise Architect и Visual Paradigm более эффективные и продвинутые по сравнению с AgroUML,так как они полностью завершены и поддерживают стандарт UML 2.0. Однако AgroUML является бесплатным, кроссплатформенным и открытым, что существенно упрощает доступ к нему. Тем не менее его незавершенность, а также отсутствие оптимальных поддержек многих диаграмм не позволяет использовать его как основное средство для проектирования диаграмм.

На основании рассмотренных вышесуществующих средств верификации можно предположить, что они не являются полноценными и не предоставляют полной картины корректности диаграммы. В связи с этим предлагается новое средство верификации UMLTester, которое является бесплатным, обеспечивает верификацию несколькими методами и позволяет максимально точно оценить корректность диаграммы.

Список использованных источников

1. *Режим* доступа: http://argouml.tigris.org/.
2. *Режим* доступа: http://www.visual-paradigm.com/.
3. *Макгрегор Дж.* Тестирование объектно-ориентированного программного обеспечения : практическое пособие / Дж. Макгрегор, Д. Сайкс ; пер. с англ. – К. : ООО «ТИД ”ДС“», 2002. – 432 с.
4. *Хоар Ч.* Взаимодействующие последовательные процессы / Ч. Хоар ; пер. с англ. – М. : Мир, 1989. – 264 с.

УДК 004.82(045)

А.І. Вавіленкова, канд. техн. наук
Національний авіаційний університет, м. Київ, Україна

ПРОГРАМНЕ ЗАБЕЗПЕЧЕННЯ ДЛЯ ВИЯВЛЕННЯ ТЕКСТОВИХ ДОКУМЕНТІВ, ІДЕНТИЧНИХ ЗА ЗМІСТОМ

А.И. Вавиленкова, канд. техн. наук
Национальный авиационный университет, г. Киев, Украина

ПРОГРАММНОЕ ОБЕСПЕЧЕНИЕ ДЛЯ ВЫЯВЛЕНИЯ ТЕКСТОВЫХ ДОКУМЕНТОВ, ИДЕНТИЧНЫХ ПО СМЫСЛУ

A.I. Vavilenkova, Candidate of Technical Sciences
National Aviation University, Kyiv, Ukraine

SOFTWARE FOR DETECTION OF TEXT DOCUMENTS IDENTICAL IN CONTENT

Проаналізовано основні методи, що лежать в основі відкритого програмного забезпечення з виявлення дублікатів електронних документів, зазначено їх недоліки: відсутність компоненти семантичного та змістовного аналізу текстів. Запропоновано систему автоматизованого формування логіко-лінгвістичних моделей як допоміжний механізм вилучення змісту з речень природної мови, на основі якого можна вирішити проблему екстракції знань з текстової інформації.

Ключові слова: *текстові документи, семантичний аналіз, логіко-лінгвістичні моделі, аналіз тексту, дублікати, природна мова.*

Осуществлен анализ основных методов, которые лежат в основе открытого программного обеспечения по выявлению дубликатов электронных документов, определены их недостатки: отсутствие компоненты семантиче-