ПОСТРОЕНИЕ TTCN ТЕСТОВ НА ОСНОВЕ UML-ДИАГРАММ

А. Ю. Подъячев

Санкт-Петербургский институт информатики и автоматизации РАН

СПИИРАН, 14-я линия ВО, д. 39, Санкт-Петербург, 199178

<gedo@yandex.ru>

УДК 004.05

Подъячев А.Ю. **Построение TTCN тестов на основе UML-диаграмм** // Труды СПИИРАН. Вып. 4. — СПб.: Наука, 2007.

Аннотация. *Статья рассматривает использование различных типов графического представления нотации UML в качестве основы для построения сценариев тестирования в методологии TTCN. В данном контексте проведена оценка диаграмм различных типов на предмет их применимости для автоматического тестирования в нотации TTCN* — Библ. 13 назв.

UDC 004.05

Podyachev A.Y. **UML diagram based TTCN tests implementing** // SPIIRAS Proceedings. Issue 4. — SPb.: Nauka, 2007.

Abstract. *The paper describes different graphic type's representation of UML notation using as base for TTCN methodology test scenarios creating. In this context evaluation of different kind diagram application was made in scope of they application in automation testing for TTCN notation.* — Bibl. 13 items.

1. Введение

Данная статья представляет применение технологии TTCN (Test and Test Control Notation), позволяющей производить детальные сценарии тестирования программного обеспечения, основываясь на методах графического моделирования методологии UML. TTCN является нотацией, использующейся при моделировании тестовых ситуаций в больших системах и системах требующих повышенного уровня контроля качества. Ее развитием и поддержкой занимается организация под названием TTMedal проект. Применение возможностей TTMedal проекта в методах тестирования, основанных на UML-диаграммах, осуществляется при построении сценариев автоматического тестирования в терминах нотации TTCN. Язык TTCN позволяет использовать метрики для структурной оценки тестовой спецификации для сценариев тестирования [1]. Практическое применение показывает, что UML-диаграммы в комбинации с различными техниками тестирования удовлетворяют требованиям автоматизированного тестирования при генерации сценариев. Нотация TTCN удовлетворяет решению множества задач – от написания тестовых спецификаций и написания скриптов до непосредственного тестирования.

2. Методология TTCN-3

Эффективное использование нотации TTCN достигается посредством интеграции модуля TTCN в оболочку Eclipse, что позволяет строить метаописания тестов, отображать метамодели спецификации в нотацию UML и предоставляет возможности параллельного метамоделирования.

Параллельное метамоделирование позволяет тестировать связи в параллельных процессах и их взаимодействие, строить тестовые спецификации для оценки параллельных процессов.

Одним из основных назначений нотации является написание детализированных спецификаций тестов.

Примерами практического приложения данной методологии является написание тестовых спецификаций для многих видов программных систем, включая мобильную связь (GSM, 3G, TETRA), беспроводные сети (Hiperlan/2), беспроводную телефонию DECT, широкополосные технологии (B-ISDN, ATM), платформы, основанные на CORBA, и Интернет протоколы, такие как IPv6, SIGTRAN, SIP и OSP. Последняя версия языка, TTCN версии 3 (TTCN-3), стандартизирована ETSI (ES 201 873 серии) и ITU-T (Z.140 серии).

Основными преимуществами использования языка TTCN-3 является то, что он:

- специально спроектирован для тестирования;
- является международно-стандартизованным языком описания тестовых сценариев;
- синтаксис и семантика операторов тестов TTCN-3 общедоступны и не привязаны к конкретным языкам программирования;
- тесты TTCN-3 имеют возможность метаописания разных уровней описания системы;
- в образовательных и тренировочных целях может быть рационализован и сокращен;
- дает возможность применения большинства методологий и стилей как корпоративного уровня, так и не стандартизованных.

Практическое применение языка осуществляется посредством встраивания тестовых описаний в UML диаграммы.

3. Уровни тестирования

Обнаружение дефектов происходит путем декомпозиции системы на составляющие уровни с последующим сопоставлением каждого уровня с заданными требованиями, которые фиксируются в документах, предназначенных для этого: MRD – требования разработки и SRS - спецификация [2]. Эти документы описывают взаимосвязанное поведение программной системы и ее структуру. Они могут быть описаны как обычным текстом в виде формальной записи, так и при помощи UML диаграмм в графическом представлении.

Перед проведением тестирования системы и запуском тестов на выполнение необходимо подготовить тестовое окружение на основе спецификации. Проект теста должен быть подготовлен до начала тестирования и содержать последовательности шагов и требования к окружению [3]. Технология нотации TTCN в данном контексте предлагает методы метамоделирования, которые позволяют осуществить реинженеринг.

Спецификации не должны содержать подробную информацию, например о том, какие значения должны быть использованы, но должны четко описывать цель текущего теста. Детализированная информация записывается в физические тестовые наборы, которые одновременно и являются отчетом о проведенном тестировании (рис. 1).

Труды СПИИРАН. 2007. Вып. 4. ISSN 2078-9181 (печ.), ISSN 2078-9599 (онлайн)
SPIIRAS Proceedings. 2007. Issue 4. ISSN 2078-9181 (print), ISSN 2078-9599 (online)
www.proceedings.spiiras.nw.ru

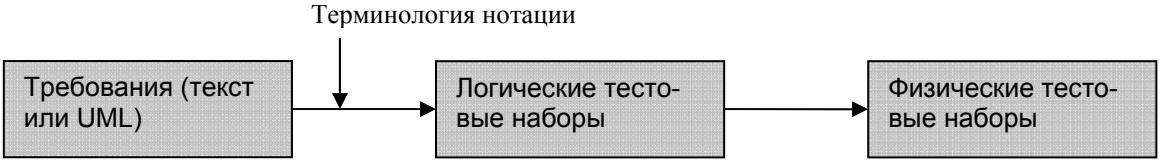


Рис. 1. Переход от требований к реализации сценария.

Предлагаемым нами подходом является метрическая оценка исходных спецификаций нотации UML [1] и отслеживание динамического изменения метрик в процессе декомпозиции тестовой модели на уровни.

4. Проектирование тестов

Исчерпывающее тестирование, как правило, невозможно в силу ограниченности времени, выделенного на тестирование. Выбор схемы тестирования должен быть сделан еще на этапе дизайна тестов. Необходимо дать заключение на основе проведенных нами оценок о том, что данный дизайн является наилучшим из возможных и способен выявить наибольшее количество дефектов в течение ограниченного промежутка времени.

Следующий рисунок показывает процесс проектирования тестового события на основе UML диаграммы с получением тестового скрипта в нотации TTCN-3.

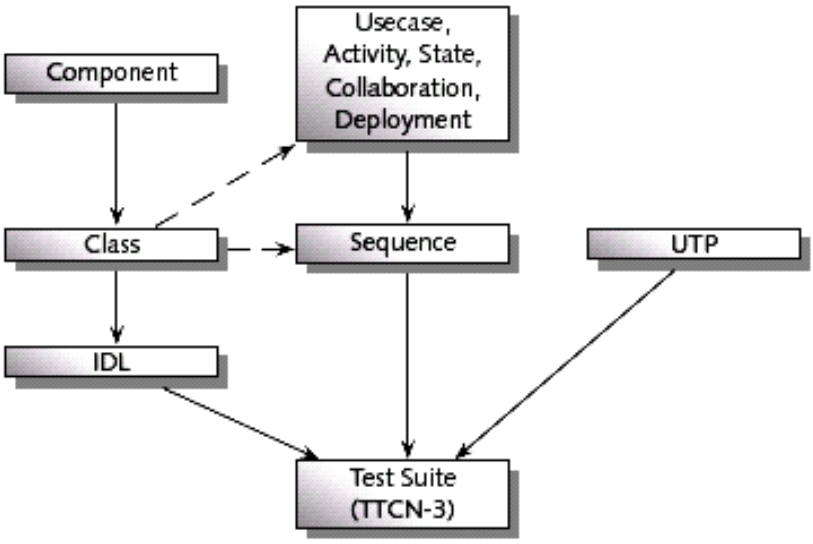


Рис. 2. Последовательность переходов при построении тестового сценария.

Информация компонентной диаграммы передается в диаграммы классов, далее при помощи диаграмм активности, состояний и взаимодействия информация передается в диаграммы последовательности. Более детальную информа-

Труды СПИИРАН. 2007. Вып. 4. ISSN 2078-9181 (печ.), ISSN 2078-9599 (онлайн)
SPIIRAS Proceedings. 2007. Issue 4. ISSN 2078-9181 (print), ISSN 2078-9599 (online)
www.proceedings.spiiras.nw.ru

цию по передаче информации между диаграммами можно получить в методах использования UML для автоматизированного тестирования [9].
По существу диаграммы активности ассоциируются с диаграммами классов и использования, и диаграммы использования и взаимодействия могут быть, в сущности, использованы для генерации нескольких диаграмм последовательности. Также диаграммы активности могут быть использованы для контроля и отслеживания выполнения тестового сценария. На последнем этапе, при помощи языка описания интерфейса (IDL - Interface Definition language) и диаграммы последовательности, в конечной реализации получается формализованный тестовый набор в нотации TTCN-3. Он содержит необходимую для выполнения теста информацию и служит для последующей декомпозиции в составляющие тестового сценария.

Некоторыми из метрических показателей, которые предлагается использовать для отслеживания тестирования и передачи модели, являются цикличеcкая сложность, размер класса и отклик класса [1].

Циклическая сложность используется для того, чтобы оценить сложность алгоритма в методе. Это число испытательных случаев, которое необходимо для всесторонней проверки этого метода. Размер класса используется, чтобы оценить простоту понимания кода разработчиками. Размер может быть измерен разными путями. Среди них подсчёт всех строк программы, числа инструкций, числа пустых строк и числа строк комментария. Отклик для класса считает набор всех методов, которые могут быть вызваны в ответ на сообщение, обращённое к объекту класса или посланное одним из методов класса. Он включает все методы, доступные в иерархии классов.

Непосредственная оценка UML диаграмм показала различные возможности их применения в построении автоматизированных тестов на этапах, показанных на рис. 2, для получения тестовых наборов. Диаграммы оценивались по следующим критериям:

- область функциональности, описываемая диаграммой;
- этап тестирования, на котором данная диаграмма может быть использована;
- степень формализации нотации UML в данном контексте;
- применимость диаграммы при проектировании автоматического теста.

Следующая таблица описывает полученный результат (табл. 1).

Таблица 1

Анализ типов диаграмм UML

Тип диаграммы	Область применения	Уровни тестирования	Формализация	Применение в а.т.
Диаграммы активности	Бизнес и рабочие процессы, процедурная логика	Модульное, интеграционное, системное	+	+
Диаграммы классов	Структура программной системы	Модульное, интеграционное, системное	+	+
Диаграммы последовательностей	Взаимодействие между пользователями и различными классами	Модульное, интеграционное, системное	+	+
Диаграммы состояний	Поведение объектов	Модульное, системное	+	+
Случаи использования (Use Cases)	Взаимодействие пользователей и программной системы	Системное	-	-

Диаграммы активности позволяют описывать процедурную логику, бизнес-логику и рабочие процессы. Эти данные используются как для низкоуровневого модульного тестирования, так и для высокоуровневого системного тестирования. Изменение состояний описывается диаграммами активности в виде условных последовательностей (if-then-else), которые подлежат тестированию, в частности, можно описать пред- и пост- условия тестирования какого-либо логического случая. Важная информация может быть получена из диаграмм классов на этапе конструирования тестов, например, типы доступных переменных и определение методов классов. Диаграммы состояний могут быть использованы для описания поведения объектов уровня системы, т.е. этот тип диаграмм может с одинаковым успехом использоваться как для модульного тестирования, так и системного.

В таблице 2 представлено сопоставление UML диаграмм с различными методами проектирования тестов.

Таблица 2

Связь диаграмм UML и методов проектирования тестов

	Эквивалентное разбиение	Анализ граничных значений	Тестирование переходов между состояниями	Использование в тестировании
Диаграммы активности	+	+		+
Диаграммы классов	++	++		
Диаграммы последовательности сообщений	+	+	+	++
Диаграммы состояний	+	+	++	
Случаи использования (Use Cases)				++

Диаграммы активности, последовательности сообщений и диаграммы состояний могут содержать информацию о поведении системы в некоторых состояниях. Эта информация применима в эквивалентном разбиении и при анализе граничных значений. Диаграммы классов гораздо меньше подходят для динамического тестирования, так как они описывают структуру системы в целом. Однако диаграммы классов в комбинации с OCL (Object Constraint Language) могут быть также использованы при разбиении и анализе граничных значений. Диаграммы состояний применяются в комбинации с методами проверки переходов между состояниями, так как и диаграммы, и методы основываются на модели состояний. Диаграммы действий, последовательности сообщений и UseCase могут описывать взаимодействие между пользователем и системой. В данном случае используется информация из сообщений собранная по диаграмме последовательности сообщений.

5. Построение сценария

Нами предлагается для определения и разработки метода генерации тестового случая принять за основу порядок сообщений. Использование порядка сообщений данной диаграммы предполагает генерацию тестового случая для каждого возможного пути через последовательность состояний описываемых диаграммой. Однако возможность использовать непосредственно диаграммы последовательности в тест-дизайнере будет регулироваться сгенерированны-

Труды СПИИРАН. 2007. Вып. 4. ISSN 2078-9181 (печ.), ISSN 2078-9599 (онлайн)
SPIIRAS Proceedings. 2007. Issue 4. ISSN 2078-9181 (print), ISSN 2078-9599 (online)
www.proceedings.spiiras.nw.ru

ми тестовыми случаями с использованием ограниченного порядка сообщений диаграммы. Здесь алгоритм непараллельного TTCN-3 выбран для того, чтобы сделать объяснение данного примера проще. Параллельный алгоритм TTCN-3 отличается от предыдущего способом генерации тестовых случаев для каждого компонента и может быть также легко применен при рассмотрении задействованных сущностей диаграммы.

Очередность в диаграммах последовательности сообщений определяется:

- общим порядком осей сущности;
- частичным порядком между сущностями;
- общим механизмом упорядочивания и области действия.

Общее упорядочивание осей сущности используется по причине необходимости упорядочивания сообщений полностью по умолчанию на одном порту. Если предпочтительно другое поведение, то порядок должен быть изменен введением соответствующих дополнительных элементов. Частичное упорядочивание между сущностями может включать некоторое упорядочивание ограниченных путей трассировки диаграммы, которые позволяют дать более точное описание тестового случая вследствие исключения нежелательных потоков.

Таким образом, частичное упорядочивание, ограниченное точками синхронизации, использует порядок отправки полученных сообщений, а графический порядок используется для принятия решения о необходимости использования дополнительных путей. Графический порядок означает упорядочивание, полученное из диаграммы слева направо и сверху вниз по диаграмме до конца или до точки синхронизации.

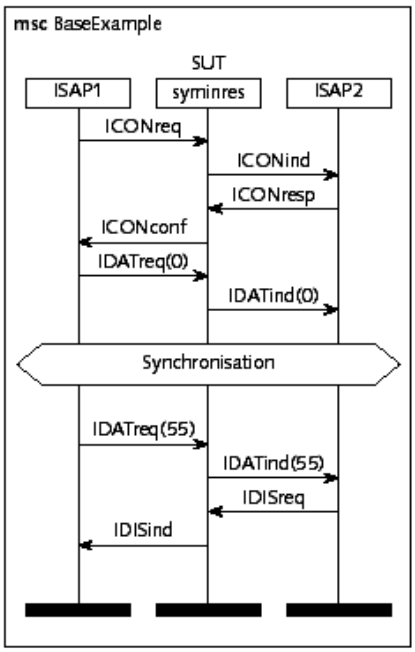
Точки синхронизации используются, чтобы четко определить, где выполняются все отправляемые и получаемые сообщения. Вследствие ограничений частичного упорядочивания между сущностями и использования преимуществ графического порядка общий механизм упорядочивания не является необходимым, но он может быть использован для дополнительной, более точной и поясняющей очередность, информации.

Область действия используется, чтобы недвусмысленно определить приоритет какого-либо упорядочивания по всем перестановкам элементов задействованных сообщений, и реализуется посредством упорядочивания операторов в TTCN-3. В случаях если правила могут быть расширены автоматическим определением порядка наложения сообщений, например при получении двух одновременных сообщений, то очередность определяется последовательностью операторов TTCN.

Синхронизация будет более точной при использовании более детальной информации, полученной по значениям текущих состояний (рис. 3).

Тестовый случай в виде TTCN-скрипта подытоживает все внутренние выражения и действия. Ссылки диаграммы сообщений синхронизируются покомпонентно и интерпретируются в выражения TTCN-3. В момент синхронизации состояний и описания локальных правил синхронизации, не должно быть никаких глобальных синхронизаций и внешних зависимостей во избежание погрешности и ошибки одновременного доступа к данным.

Труды СПИИРАН. 2007. Вып. 4. ISSN 2078-9181 (печ.), ISSN 2078-9599 (онлайн)
SPIIRAS Proceedings. 2007. Issue 4. ISSN 2078-9181 (print), ISSN 2078-9599 (online)
www.proceedings.spiiras.nw.ru



(a) MSC

```
testcase BaseExample() runs on syminres {
    ISAP1.send( ICONreq )
    ISAP2.receive( ICONind )
    ISAP2.send( ICONresp )
    ISAP1.receive( ICONconf )
    ISAP1.send( IDATreq ( 0 ) )
    ISAP2.receive( IDATind ( 0 ) )
    ISAP1.send( IDATreq ( 55 ) )
    ISAP2.receive( IDATind ( 55 ) )
    ISAP2.send( IDISreq )
    ISAP1.receive( IDISind )
}
```

(b) TTCN-3

Рис. 3. Представление диаграммы последовательности в виде тестового случая TTCN-3.

В данной схеме тестирования метрические показатели позволяют нам сопоставить метрики исходной и конечной формализации и оценить степень, в которой модель изменилась. Данная оценка проводится на уровне метаописания.

6. Заключение

Приведенная технология может применяться не только в автоматизации тестирования как прикладной инструмент, но и для исследования методов построения моделей программного обеспечения в различных нотациях и их дальнейшего использования в качестве спецификаций для построения тестовых сценариев. С использованием метрической оценки данных моделей возникает возможность контролировать сложность, надежность и качество, как исходной спецификации, так и последующих реализаций программных комплексов. На основе выводов из оценки схемы построения тестового сценария сделать выводы и дать рекомендации относительно улучшения кода либо пересмотра спецификации в случае наличия проблем с надежностью системы. Предложенный алгоритм дает возможность абстрагироваться от конкретных описаний метамоделей и позволит использовать IDEF-0 в качестве спецификации.

Литература

1. *Подъячев А. Ю., Афанасьев С. В.* Тестирование трансляции формальных моделей // Труды СПИИРАН. СПб.: Наука, 2006. Вып. 3, т. 2. С.156–161.
2. *ISO/IEC*, 9646-3, 1998 Information technology – Open systems interconnection –Conformance testing methodology and framework – Part 3: The Tree and Tabular combined Notation (TTCN).
3. Visualization of TTCN Test Cases by MSCs / *Grabowski J., Walter T.* SAM98, Humboldt University Berlin, 1998.
4. An Industrial use of FP: A tool for generating test scripts from system specifications / *Baker P., Jervis C., King D.* // Trends in Functional Programming. Scottish Functional Programming Workshop. London, 2002.

5. TTCN, SDL and MSC-based specification and automated test case generation for INAP / *Grabowski J., Hogrefe D.* // Modeling and Analysis, Nashville, March 2000.
6. *Vassiliou-Gioles T., Li M., Schieferdecker I., Born M., Winkler M.* Configuration and execution support for distributed systems. // IWTCS'99, Budapest, Hungary, Sept. 1999.
7. *The Open Group.* ADL 2.0 Translation System [Электронный ресурс] // <http://adl.opengroup.org/> (по состоянию на 10.03.2007)
8. UML Framework for Automated Generation of Component-Based Test Systems. Software Engineering Applied to Networking and Parallel / *Born M., Schieferdecker I., Li M.* Distributed Computing (SNPD'00), Reims, France, 2000.
9. Using UML for Automatic Test Generation / *Crichton C.* Conference on Automated Software Engineering, ASE'2001.
10. *Rudolph E., Schieferdecker I., Grabowski J.* HyperMSC – a Graphical Representation of TTCN. // Proc. of the 2nd Workshop on SDL and MSC (SAM'2000), Grenoble (France), June, 26 – 28, 2000.
11. *Testing Technologies.* TT Tool Series. [Электронный ресурс] // <http://www.testingtech.de/products> (по состоянию на 10.03.2007).
12. *Канер С., Фолк Д., Нгуен Е.К.* Тестирование программного обеспечения. ДиаСофт, 2000. 544 с.
13. *Дастин Э., Рэшка Д., Пол Д.* Автоматизированное тестирование программного обеспечения. Лори, 2003 г. 592 с.