УДК 004.415

И.В. Никифоров, П.Д. Дробинцев, В.П. Котляров

ИНТЕГРАЛЬНЫЕ КРИТЕРИИ ПРОВЕРКИ ТРЕБОВАНИЙ К ПРОГРАММНОМУ ОБЕСПЕЧЕНИЮ

I.V. Nikiforov, P.D. Drobintsev, V.P. Kotliarov

INTEGRATED CRITERIA OF SOFTWARE REQUIREMENT CHECKING

Описаны методы разработки критериев проверки требований, которые позволяют обеспечить стопроцентное покрытие тестовыми сценариями исходных спецификаций на систему. Предложено использовать подход критериальных цепочек, используемых в матрице отслеживания покрытия. По критериальным цепочкам в автоматическом режиме создаются тестовые сценарии.

Изучены методы генерации сценариев и отбор сценариев, удовлетворяющих выбранному интегральному критерию покрытия.

Использование предложенных интегральных критериев проверки требований на этапе разработки формальной спецификации системы позволяет покрыть тестовыми сценариями все исходные требования системы.

ДИЗАЙН СПЕЦИФИКАЦИЙ. ТРЕБОВАНИЯ. МАТРИЦА ОТСЛЕЖИВАНИЯ. ГИДЫ.

The paper describes developing methods of tests for the requirements checking that provide 100 % coverage of the initial specifications for the system by test cases. Proposed approach of criterion chains are used in the traceability matrix. The criteria chains are used to automatic generation of test scenarios. Also the article describes methods for generating scenarios and selection of scenarios that satisfy the selected integral coverage criteria.

The use of suggested integrated tests criteria for the requirements checking at the stage of a formal specification of the system, allows to cover by test scenarios all the initial requirements of the system.

DESIGN SPECIFICATION. REQUIREMENTS. TRACEABILITY MATRIX. GUIDES.

Среди основных проблем автоматизации разработки и тестирования программного обеспечения промышленных приложений в виде сосредоточенных модулей или распределенных сетей непременно отмечается проблема обработки сложных и больших по объему спецификаций требований. Документы, фиксирующие спецификации требований, пишутся, как правило, на естественном языке и могут содержать сотни и тысячи пунктов требований. В силу этого задача формализации требований для описания поведенческих сценариев, используемых для разработки автоматических тестов или ручных тестовых процедур, характеризуется как задача огромной сложности и трудоемкости.

Применимость формальных методов в промышленности в огромной степени определяется тем, насколько адекватен язык формализации принятой инженерной практике, в которую вовлечены не только собственно разработчики кода и тестировщики, но и заказчики, руководители проектов разных уровней, маркетологи и другие специалисты. Ясно, что никакой чисто логический язык не подходит для адекватной формализации требований, которая бы одновременно сохранила семантику разрабатываемого приложения и удовлетворяла всех «причастных лиц» [1].

В современной проектной документации формулировка исходных требований задается либо конструктивно, когда из текста требования на естественном языке удается реконструировать процедуру контроля или сценарий проверки выполнения данного требования, либо неконструктивно, когда заданное в требовании свойство не содержит пояснения способа его проверки.

Проверка выполнения требований. Процедура проверки требования – это точная

последовательность причин и следствий некоторых активностей (кодируемых действиями, сигналами, состояниями), в результате анализа которой можно утверждать, что данное требование выполнено или нет. Подобная процедура проверки может использоваться в качестве критерия выполнения конкретного требования, т. е. стать критериальной процедурой. В дальнейшем изложении для критериальной процедуры будем использовать термин *последовательность* или «*цепочка*» *событий*.

Отслеживая в поведенческом сценарии системы (гипотетическом, реализованном в модели или в реальной системе) факты выполнения критериальной процедуры, можно утверждать, что соответствующее требование в анализируемой системе удовлетворено.

Процедура проверки требования (цепочка) формулируется путем задания для всех элементов цепочки следующей информации:

условий (причин), требующихся для активизации некоторой активности;

самой активности, подлежащей исполнению при данных условиях;

следствий – наблюдаемых (измеряемых) результатов исполнения указанной активности.

Для описания причин и следствий используются сигналы, сообщения или транзакции, обычные в коммуникациях инстанций реактивной системы [2], а также состояния переменных в виде значений или ограничений на области допустимых значений. Отслеживая изменения в состояниях, производимых активностями цепочек, можно наблюдать за покрытием соответствующих цепочек. При анализе допустимо рассматривать прямой переход из состояния в состояние с пустой активностью, а в случае недетерминизма – альтернативные варианты изменения состояний.

Проблемы неконструктивного задания требований преодолеваются в процессе разработки процедур проверки выполнения требований на пользовательских или межкомпонентных интерфейсах. То есть сформулированное требование о наличии некоторой функциональности или свойства надо проинтерпретировать на функциях пользовательского интерфейса или API (Application Program Interface) компонент, описывающих или именующих закодированную функциональность так, чтобы на основе этой интерпретации можно было задать процедуру проверки наличия соответствующей функциональности в приложении.

Таким образом, критерием выполнения требований могут служить цепочки, содержащие последовательности активностей и состояний; помимо этого возможны случаи, в которых критерий выполнения некоторого требования задается не одной, а несколькими цепочками.

Исходные документы, фиксирующие требования к приложению. При формулировке технических требований в промышленных проектах обычно используются следующие виды документов:

- Marketing Requirement Specification (MRS) – перечисление новой функциональности с точки зрения ее потребления пользователем, часто описываемой в виде пользовательского сценария;
- Technical Requirement Specification (TRS) – перечисление всей функциональности, подлежащей разработке, с кратким уточнением каждой функции;
- Functional Requirement Specification (FRS) – детальное описание всей функциональности, подлежащей разработке, достаточное для разработки тестовых наборов для проверки программного продукта на всех этапах жизненного цикла.

Формулировка каждого требования в этих документах чаще всего осуществляется на естественном языке и выражается одним из двух способов:

в виде поведенческого требования, когда из текста требования на естественном языке удается реконструировать сценарий (процедуру) выполнения данного требования;

в виде неповеденческого требования, задающего состав, структуру или пожелание наличия некоторого свойства без пояснения способа его потребления (способа проверки).

Поведенческие требования в силу конструктивности своего задания допускают

для проверки своей реализованности статические и/или динамические методы верификации и тестирования.

Неповеденческие требования по определению заданы неконструктивно и требуют дополнительной информации для воссоздания конструктивного сценария их проверки, чтобы верификация и тестирование стали применимы.

Формализация любых конструктивно заданных требований возможна, как возможен и эффективный автоматизированный анализ требований к программному изделию, и реализована в технологии VRS/TAT [3].

В технологии VRS/TAT для высокоуровневого описания модели, в рассматриваемой технологической цепочке используется нотация Use Case Maps (UCM) [4], а инструменты автоматизации проверки и генерации работают с моделью на языке базовых протоколов [5].

Матрица отслеживания. Формализация требований верификационного проекта начинается с разработки матрицы отслеживания TRM – Traceability matrix (на рис. 1 TRM для конкретного проекта представлена в виде таблицы). Столбцы 4 и 6 содержат идентификатор требования, употребляемый в TRS или FRS, а также полный текст требования, подлежащий формализации.

Следующий важный этап создания TRM – разработка критериальных цепочек для требований. Примеры цепочек представлены в колонке 8 «Scenario of requirement verifying». Например, в строке 291 цепочка для покрытия требования TRS_18081-2209 обозначена в колонке 10 «Scenario ID» как scen#045 и состоит из сценария scen#064, за которым исполняются еще две указанные активности. Для покрытия требования TRS_18081-2212 (строка 293) в столбце 8 указаны две цепочки: первая для автоматического, вторая для ручного тестирования. В требовании TRS_18081-2209 использованы сообщения «Do Not Disturb» и «My Avaiability changed to Do Not Disturb», появляющиеся на экране мобильного телефона. Информация о том, в каком месте экрана они появляются и с какими активностями связаны, извлечена из дополнительной технической документации, описывающей пользовательский интерфейс телефона. Следует заметить, что при создании критериальных цепочек строится модель верифицируемой функциональности, в процессе чего вводится много переменных состояния, типов, агентов, инстанций и т. п.

Разработка интегральных критериев покрытия требований. Выше отмечалась ха-

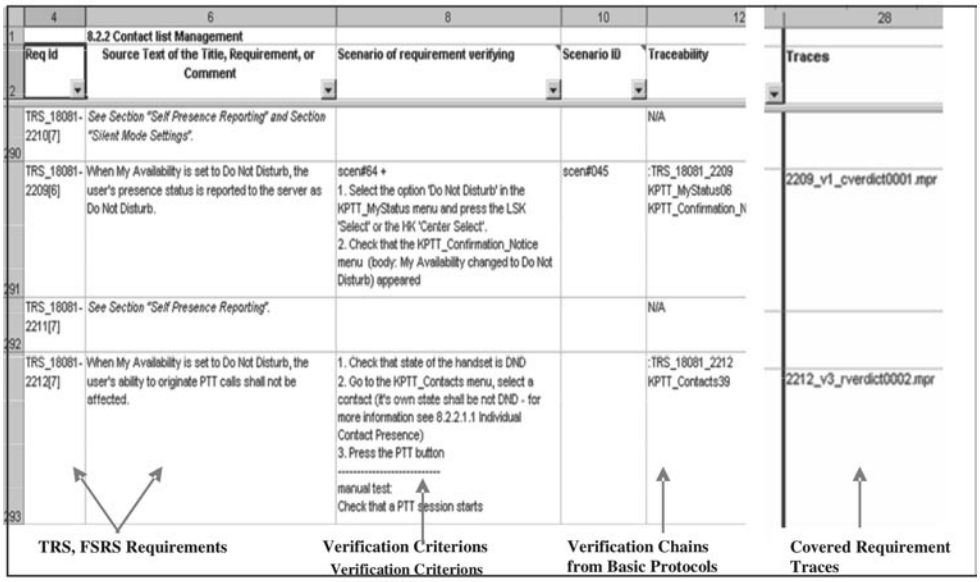

Рис. 1. TRM-матрица отслеживания

рактерная черта технологии VRS/TAT – наличие специальных критериев проверки покрытия для каждого требования. Перечислим критерии, относящиеся к требованиям, по степени роста их мощности (задача каждого критерия – стопроцентное покрытие требований):

- степень покрытия сгенерированными сценариями подмножества событий, использованных в критериальных цепочках;
- степень покрытия сгенерированными сценариями подмножества цепочек (состоящих их событий и состояний переменных), не менее чем по одной для каждого требования;
- степень покрытия сгенерированными сценариями всего множества цепочек, задающего интегральный критерий покрытия требований.

Критерии должны применяться гибко и могут изменяться в зависимости от условий генерации сценариев.

Для высокоуровневого описания модели, в рассматриваемой технологической цепочке используется нотация UCM (рис. 2), а инструменты автоматизации проверки и генерации работают с моделью на языке базовых протоколов.

Генерация сценариев и отбор сценариев, удовлетворяющих выбранному интегральному критерию покрытия. Генерация трасс осуществляется символьным и конкретным трассовыми генераторами STG (SimbolicTrace Generator) и CTG (Concrete Trace Generator), реализующими эффективные алгоритмы проверки моделей (Model Checking). Основной проблемой трассовой генерации является «взрыв» вариантов перебора при генерации сценариев (трасс) из BP, которые формализуют сценарные события, условия их реализации и соответствующее изменение состояния модели после реализации. Инструментом решения является фильтрация вариантов генерации на основе многочисленных ограничений, специально выбираемых перед циклом генерации трасс [6]: ограничения на число конечных (Goal) и промежуточных (Visited) состояний; ограничения на максимальное количество протоколов, используемых в трассе; максимальное количество трасс, генерируемых в одном цикле генерации.

Специальные ограничения задаются последовательностями событий UCM модели, направляющими процесс генерации в интересующем пользователя поведении модели (т. н. гиды – Guides). Используется два этапа для генерации тестовых сценариев по гидам. На первом этапе на основе UCM модели создаются гиды, обеспечивающие заданные критерии покрытия поведения системы. На втором этапе гиды в нотации UCM (рис. 3 *а*) преобразуются в гиды на языке базовых протоколов (рис. 3 *б*), и под их управлением производится генерация трасс (рис. 3). Важно отметить, что в гидах фиксируются лишь главные контрольные точки поведения, в то время как трасса, полученная по гиду, содержит детальную последовательность элементов поведения.

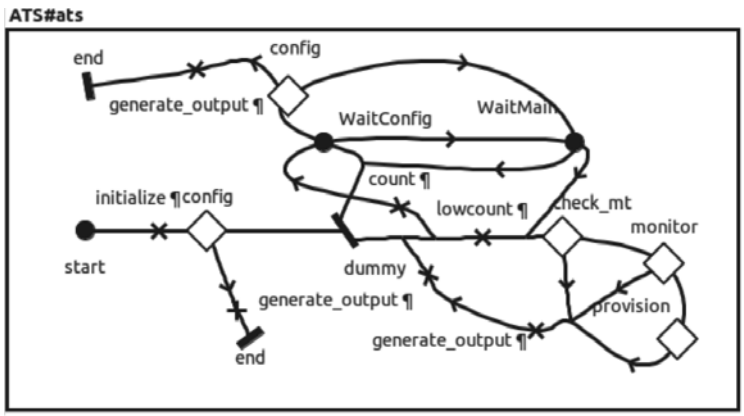

Рис. 2. Пример описания поведенческой модели на языке UCM

а)

б)

```
patterns: obj(
    p_Guide:(
        ats_0(initialize#1);
        ats_0(timer_exp_1#1);
        ats_0(load_config_1#1);
        ats_0(set_timer_1#1);
        ats_1(lowcount#1);
        ats_1(dummy#1);
        ats_1(report_req_SADM#1);
        ats_2(generate_output_2#1)
    )
);
guides: obj(
    Number_Of_Traces:1;
    g_Guide:(
        pattern:(p_Guide)
    )
);
```

Рис. 3. Гиды: *а* – в нотации UCM; *б* – на языке Guide Language VRS

Такой подход к генерации существенно уменьшает влияние комбинаторного взрыва на время генерации трасс при обходе дерева поведения проектируемой системы.

Результаты применения в пилотных проектах. В таблице приведены результаты применения интегрированной технологии проектирования и тестирования в области разработки беспроводных телекоммуникационных приложений. В результате получено существенное снижение трудозатрат и увеличение качества программного продукта.

Результатом работы является усовершенствованная технология, интегрирующая верификацию и тестирование программных проектов, которая обеспечивает:

полную автоматизацию процесса разработки промышленного программного продукта с контролем реализации семантики требований;

генерацию модели приложения и символьных поведенческих сценариев, стопроцентно покрывающих поведенческие свойства приложения;

Результаты применения технологии в пилотных проектах

Проект	Количество требований	Количество базовых протоколов	Степень покрытия требований, %	Количество найденных и исправленных ошибок (всего/существенных)	Трудоемкость (человеко-неделя)
Модуль 1 беспроводной сети (БС)	400	127	75	142/11	5,6
Модуль 2 БС	730	192	80	106/18	12
Модуль верхнего уровня БС	148	205	100	68/23	11
Модуль связи клиентов и администратора БС	106	163	100	42/8	6
Модуль ПО мобильного телефона	200	170	100	96/10	7
Модуль интеграции в системе управления автомобилем	2220	1533	65	573/57	17
Радиоприемник автомобиля	26	26	100	14/4	3

автоматическую конкретизацию символьных трасс в соответствии с планом тестирования;

высокую автоматизацию процесса разработки и управления качеством программного продукта.

СПИСОК ЛИТЕРАТУРЫ

1. **Баранов, С.** Индустриальная технология автоматизации тестирования мобильных устройств на основе верифицированных поведенческих моделей проектных спецификаций требований [Текст] / С. Баранов, В. Котляров, А. Летичевский // Тр. Междунар. науч. конф. Космос, астрономия и программирование. —СПб.: Изд-во СПбГУ, 2008. —С. 134—145.

2. **Manna, Z.** The Temporal Logic of Reactive and Concurrent Systems [Text] / Z. Manna, A. Pnueli. —Springer-Verlag, 1992.

3. **Baranov, S.** The technology of Automation Verification and Testing in Industrial Projects [Электронный ресурс] / S. Baranov, V. Kotlyarov, A. Letichevsky, P. Drobintsev // Proc. of St. Petersburg IEEE Chapter, International Conf. May 18-21, 2005. —P. 81—86.

4. Recommendation ITU-T Z.151 [Электронный ресурс] / User requirements notation (URN), 11/2008.

5. **Letichevsky, A.** Basic Protocols, Message Sequence Charts, and the Verification of Requirements Specifications [Text] / A. Letichevsky, J. Kapitonova, A. Letichevsky Jr., V. Volkov, S. Baranov, V. Kotlyarov, T. Weigert // Proc of ISSRE04 Workshop on Integrated-reliability with Telecommunications and UML Languages (ISSRE04:WITUL). —IRISA Rennes France, 02.11.2004.

6. **Колчин, А.** Направленный поиск в верификации формальных моделей [Текст] / А. Колчин // Тези доп. міжнар. конф. Теоретичні та прикладні аспекти побудови програмних систем TAAPSD'2007. — Бердянск: НаУКМА, Національний ун-т ім. Т.Г. Шевченка, Ін-т програмних систем НАН України, 2007. — С. 256—258.

REFERENCES

1. **Baranov S., Kotliarov V., Letichevskii A.** Industrial'naia tekhnologiia avtomatizatsii testirovaniia mobil'nykh ustroistv na osnove verifitsirovannykh povedencheskikh modelei proektnykh spetsifikatsii trebovanii / Trudy mezhdunar. nauch. konf. Kosmos, astronomiia i programmirovanie. — St Petersburg: Izd-vo SPbGU, 2008. — S. 134—145. (rus)

2. **Manna Z., Pnueli A.** The Temporal Logic of Reactive and Concurrent Systems. — Springer-Verlag, 1992.

3. **Baranov S., Kotlyarov V., Letichevsky A., Drobintsev P.** The technology of Automation Verification and Testing in Industrial Projects / Proc. of St. Petersburg IEEE Chapter, International Conf., May 18-21, St. Petersburg, 2005. — P. 81—86. (rus)

4. Recommendation ITU-T Z.151. User requirements notation (URN), 11.2008.

5. **Letichevsky A., Kapitonova J., Letichevsky A. Jr., Volkov V., Baranov S., Kotlyarov V., Weigert T.** Basic Protocols, Message Sequence Charts, and the Verification of Requirements Specifications. Proc of ISSRE04 Workshop on Integrated-reliability with Telecommunications and UML Languages (ISSRE04: WITUL). — IRISA Rennes France, 02.11.2004.

6. **Kolchin A.** Napravlennyi poisk v verifikatsii formal'nykh modelei // Tezi dop. mizhnar. konf. «Teoretichni ta prikladni aspekti pobudovi programnikh sistem TAAPSD'2007». — Berdiansk: NaUKMA, Natsional'nii un-t im. T.G. Shevchenka, In-t programnikh sistem NAN Ukraïni, 2007. — S. 256—258.

НИКИФОРОВ Игорь Валерьевич — *аспирант кафедры информационных и управляющих систем Санкт-Петербургского государственного политехнического университета.*
195251, Россия, Санкт-Петербург, ул. Политехническая, д. 29.

NIKIFOROV, Igor V. *St. Petersburg State Polytechnical University.*
195251, Politekhnicheskaya Str. 29, St. Petersburg, Russia.

ДРОБИНЦЕВ Павел Дмитриевич — *доцент кафедры информационных и управляющих систем Санкт-Петербургского государственного политехнического университета, кандидат технических наук.*
195251, Россия, Санкт-Петербург, ул. Политехническая, д. 29.

DROBINTSEV, Pavel D. *St. Petersburg State Polytechnical University.*
195251, Politekhnicheskaya Str. 29, St. Petersburg, Russia.

КОТЛЯРОВ Всеволод Павлович – *профессор кафедры информационных и управляющих систем Санкт-Петербургского государственного политехнического университета, кандидат технических наук.*

195251, Россия, Санкт-Петербург, ул. Политехническая, д. 29.
E-mail: vpk@spbstu.ru

KOTLYAROV, Vsevolod P. *St. Petersburg State Polytechnical University.*
195251, Politekhnicheskaya Str. 29, St. Petersburg, Russia.
E-mail: vpk@spbstu.ru