УДК 004.4’2

В. П. Котляров, П. Д. Дробинцев

Санкт-Петербургский государственный политехнический университет
ул. Политехническая, 29, Санкт-Петербург, 195251, Россия

E-mail: vpk@spbstu.ru

АВТОМАТИЗАЦИЯ ПРОЕКТИРОВАНИЯ ПРОГРАММНОГО ПРОДУКТА С ПОМОЩЬЮ ФОРМАЛЬНЫХ СПЕЦИФИКАЦИЙ [*]

Рассматривается проблема ручной разработки формальных спецификаций создаваемого приложения, вносящая существенную часть трудоемкости в производственный цикл, совместно с проблемой контроля семантики приложения в соответствии с семантикой заказанной заказчиком. Обсуждаются средства автоматизации разработки формальных спецификаций моделей требований и моделей архитектур и технология проектирования спецификаций и моделей в нотации, доступной для контроля заказчика.

Ключевые слова: модели требований, архитектурные модели, семантика требований, контроль заказчика.

Введение

Проблема полной автоматизации цикла разработки современного программного продукта крайне актуальна. Ее решение позволяет повысить производительность и качество создаваемого продукта. В настоящее время существуют позитивные примеры подобных промышленных технологий [1], где по разработанным вручную спецификациям автоматически генерируются коды приложений и коды тестов. Однако при всех успехах автоматической генерации кода автоматизация разработки исходных формальных спецификаций все равно остается трудоемкой, рутинной работой, занимающей существенную часть производственного цикла. С другой стороны, существует проблема контроля соответствия семантики разрабатываемого приложения семантике заказанной заказчиком.

Целью настоящей работы является обсуждение технологии и средств:

- автоматизации разработки формальных спецификаций моделей требований и моделей архитектур создаваемых приложений;
- осуществления проектирования спецификаций и моделей в нотации, доступной для контроля заказчика и принятия с заказчиком согласованных решений по семантике автоматически генерируемого кода.

Формализация моделей требований

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

[*] Работа поддержана грантом РФФИ 11-07-90412_Укр_ф_а.

[1] IBM: Лучшие средства разработки программного обеспечения: http://www-01.ibm.com/software/ru/rational/software_development

	A	B	C	D	E
1	ID	Identifier	Requirements	Scenario of Requirements	Traceability
2	1	1639	When the CCGW is ready to send the convertional channe out-of-service ICP message to the ZC to take the channel out of service, it shall send a proper STOP ICW message to the BS before it stops sending the link management packets. Note: This is when an active call is in progress. Transmission of STOP IMBE message is optional	1) Receive CChannelDown 2) Send XisStopBS 3) stop link 4) Send ChannelOutOfServiceRequest	:1639 rChannelDown1

Рис. 1. Требование, цепочка (сценарий) и имя соответствующего BP, кодирующего цепочку

Поведенческие требования, при наличии описанного сценария выполнения, являются конструктивно заданными и допускают для проверки реализуемости использование верификации или тестирования. Неповеденческие требования часто задаются неконструктивно, что заставляет в процессе формализации привлекать дополнительную информацию, позволяющую реконструировать сценарий их проверки, т. е. привести неконструктивную форму задания требования к конструктивной.

Процедура или сценарий проверки некоторого требования заключается в задании перечислимого упорядоченного набора событий, фиксация которого в процессе исполнения приложения будет для заказчика критерием выполнимости соответствующего требования. Подобную последовательность событий в дальнейшем изложении будем называть критериальной последовательностью или цепочкой [1]. Каждому требованию может быть сопоставлено одна или несколько критериальных цепочек. Отслеживая в поведенческом сценарии системы факт выполнения критериальной цепочки, можно утверждать, что соответствующее требование в анализируемой системе удовлетворено.

Специальная структура данных (Traceability Matrix) проектирует исходные требования (колонка Requirements) на критериальные цепочки (колонка Scenario for the Requirement). Каждая цепочка содержит события, выполнимость которых необходима для покрытия некоторого требования (например, требования 1 639 на рис. 1). В процессе формализации этот сценарий кодируется базовым протоколом rChannelDown1 (колонка Traceability) [2].

Базовый протокол (BP) кодирует минимальный наблюдаемый шаг поведения системы. BP – это аналог тройки Хора, содержащий предусловие, постусловие и процесс (наблюдаемое действие, последовательность действий). Пред- и постусловия описывают подмножество состояний системы перед и после действий процесса, заключающегося в посылке сигналов или изменения значений переменных приложения. BP может содержать символьные или конкретные значения параметров, причем для символьных задается диапазон возможных значений. BP может относиться к одному или нескольким требованиям, равно как и несколько BP или несколько цепочек BP могут относиться к одному требованию.

Множество BP составляет модель требований. Объединяя непротиворечивые пред- и постусловия различных BP, можно построить сценарии или трассы, позволяющие наглядно отображать возможные поведения проектируемой системы. Трассы, использующие символьные параметры, называются символическими. Корректность варианта поведения, фиксируемого конкретной или символической трассой, доказывается верификатором [3; 4].

Совокупность трасс, покрывающая все цепочки всех требований, формирует модель требований разрабатываемого приложения. Эта модель может использоваться для генерации полного набора тестов, проверяющего функциональное покрытие всех конструктивно заданных требований.

Формализация архитектурной модели

Формальная модель приложения на основе BP обладает серьезным недостатком: она содержит все возможные трассы, которые можно построить из BP. Даже для индустриальных проектов среднего размера это обстоятельство приводит к взрыву вариантов, которые следу-

ет учитывать при анализе. В то же время в приложениях для заказчика важны не все поведенческие трассы, а только те, которые можно будет использовать для реализации функциональности, заданной требованиями [5].

Модель, отображающую реализацию функциональности, называют архитектурной. Она часто задается в нотации Use Case диаграмм [2]. UCM-диаграммы изображают причинно-следственные связи событий (Responsibilities) на путях от начальной (Start point) до конечной (End point) точки. Каждое событие (Responsibility) на пути UCM-диаграммы может быть закодировано BP. Путь может разветвляться в точках альтернатив (OrFork) или точках распараллеливания (AndFork), ветви могут соединяться в точках (OrJoin и AndJoin) и, если необходимо, синхронизироваться. В путях могут использоваться таймеры (Timer) и поддиаграммы (Stub).

Например, последовательность событий, описывающих поведение системы, задается набором взаимодействующих между собой диаграмм (рис. 2).

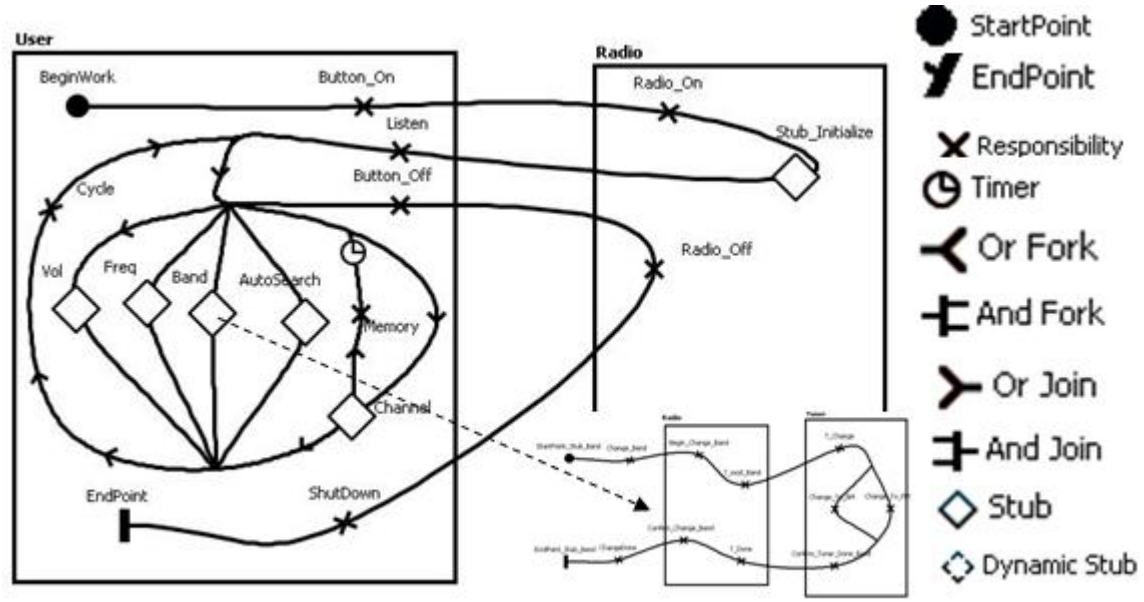

Рис. 2. UCM дает наглядное представление поведения системы и взаимодействий между ее компонентами

В настоящее время UCM представляет наиболее высокоуровневое описание проектируемой системы, сохраняющее при этом все сценарии ее поведения, она понятна и проста. Высокоуровневое описание можно многократно детализировать, добиваясь ясности и корректности описания архитектуры системы.

Эта форма описания позволяет представителю заказчика участвовать в принятии решений на уровне архитектуры проекта, в текущем контроле проекта, в отслеживании расхождений семантики постановки задачи и ее реализации.

Пример проектирования телекоммуникационного приложения

Диаграмма UCM представляет поведение системы (рис. 3), разработанной на основе стандарта CDMA (Code Division Multiple Access) и включающей такие модули, как MS – Mobile Subscriber (1) – мобильное устройство, претендующее на проведение процедуры handover; SDU – Selection Distribution Unit (2) – модуль, предоставляющий функциональность SHO

[2] Recommendation ITU-T Z.151 User requirements notation (URN) – Language Definition. 2008.

(Soft Hand Off) для осуществления handover; MM – Mobility Manager (3) – модуль, управляющий процессом handover; MSC – Mobile Switching Center (4) – модуль, управляющий информацией, связанной с мобильным устройством; BTS – Base Transmitter Station (5) – базовая станция, осуществляющая прием информации от мобильного устройства. Часть из представленных модулей парные, при этом приставка s (*source*) означает принадлежность к текущему оборудованию, которое производит обслуживание, а t (*target*) – принадлежность к целевому оборудованию, к которому будет произведено подключение устройства после осуществления handover. Функция handover – с передачей голоса или информации, в данном сценарии осуществляется handover с передачей информации (г) после успешного определения типа handover.

Сценарий рис. 3 (отмечен пунктиром) описывает успешное проведение процедуры handover. Мобильное устройство посылает запрос MyRequestHandoff на модуль SDU (*а*), который распознает запрос и перенаправляет его посредством сигнала C10HandoffRecognized модулю MM (*б*), находящемуся в области действия мобильного устройства, который в свою очередь посылает данные A1HandoffRequired (voice) модулю MSC (*в*), обеспечивающему поиск целевого модуля MM и передачу запроса ему. Целевой модуль MM параллельно опрашивает целевой SDU на наличие свободного ресурса C3SDFResourceRequest и определяет тип требуемого и наличия свободного ресурса, для проведения handover устройству выделяется канал C3SDFResourceResponse(DetailedCause) (*д*), и квитанция C10TargetChannelReady о его выделении отправляется мобильному устройству через модули MM, MSC и SDU (*е*). После получения данных о канале мобильное устройство начинает процесс передачи данных в рамках handover (сигналы CAIHandoffCompletion и Speech), а также начинает подготовку к освобождению занятых ресурсов (*з*). После завершения передачи данных целевой модуль MM посылает сигнал завершения handover (A1HandoffComplete), который позволяет начать процедуру освобождения ресурсов (*и*). Признаком окончания освобождения ресурсов является получение модулем MSC сигнала A1ClearComplete, который и завершает процедуру handover (*к*).

Сценарий рис. 3 избыточно детализирован, что препятствует простоте и наглядности его анализа. Разбиение целого сценария на фрагменты (рис. 4) и абстрагирование деталей поведения элементами Stub (рис. 5, 6) позволяет решить эту проблему.

Stub на рис. 5 абстрагирует функциональность определения типа handover и запроса ресурса, раскрытую на рис. 4.

Поддиаграмма Stub (рис. 6) описывает следующее поведение.

- Параллельный запрос ресурса и начало процедуры определения типа handover с передачей голоса или без передачи голоса.
- В рамках запроса ресурса сигнал C3SDFResourceRequest посылается на модуль SDU, который отвечает сигналом об успешном выделении ресурса C3SDFResourceResponse (DetailedCause).
- В случае handover с передачей голоса посылается запрос на использование ресурсов модуля VPU (C8VocoderResourceRequest).
- В случае handover без передачи голоса – запуск таймера ожидания выделения ресурса от модуля SDU.
- В случае удачного выделения ресурсов в заданные таймером временные интервалы посылается сигнал C10SDFHardHandoffChannelAssigned (voice, circuit) на модуль SDU, сигнализирующий об успешном выделении ресурсов и выдаче канала для handover.

Созданная архитектурная модель согласуется с представителем заказчика и служит основой для генерации модели базовых протоколов из UCM-описания.

Автоматический цикл проектирования приложения

Проектирование UCM архитектурных моделей упрощает процесс и трудоемкость проектирования исходных формальных спецификаций, но остается ручной работой. Зато последующие фазы создания программного продукта полностью автоматизированы.

На рис. 7 приведены основные преобразования создаваемого приложения. На первом этапе исходные требования на неформальном языке вручную преобразуются в нотации UCM.

Исходные требования на систему могут быть представлены как в виде неформального текста, так и в виде полуформальных описаний, например, таблиц или различного рода диаграмм. В рамках формализации выделяются основные сущности системы, от переменных которой будет зависеть ее поведение, а также входной и выходные интерфейсы. Вся полученная информация служит для создания множества ВР, которые используются в дальнейшем как для генерации кода модели поведения системы, так и для генерации тестов.

На втором этапе по созданной спецификации автоматически генерируется модель базовых протоколов, которая позволяет верифицировать проектную спецификацию. Проверке подлежат такие свойства системы, как полнота и непротиворечивость. Проверка осуществляется в процессе генерации возможных последовательностей ВР и доказательства корректности их применимости. Сопровождающий этот процесс взрыв вариантов ограничивается фильтрами

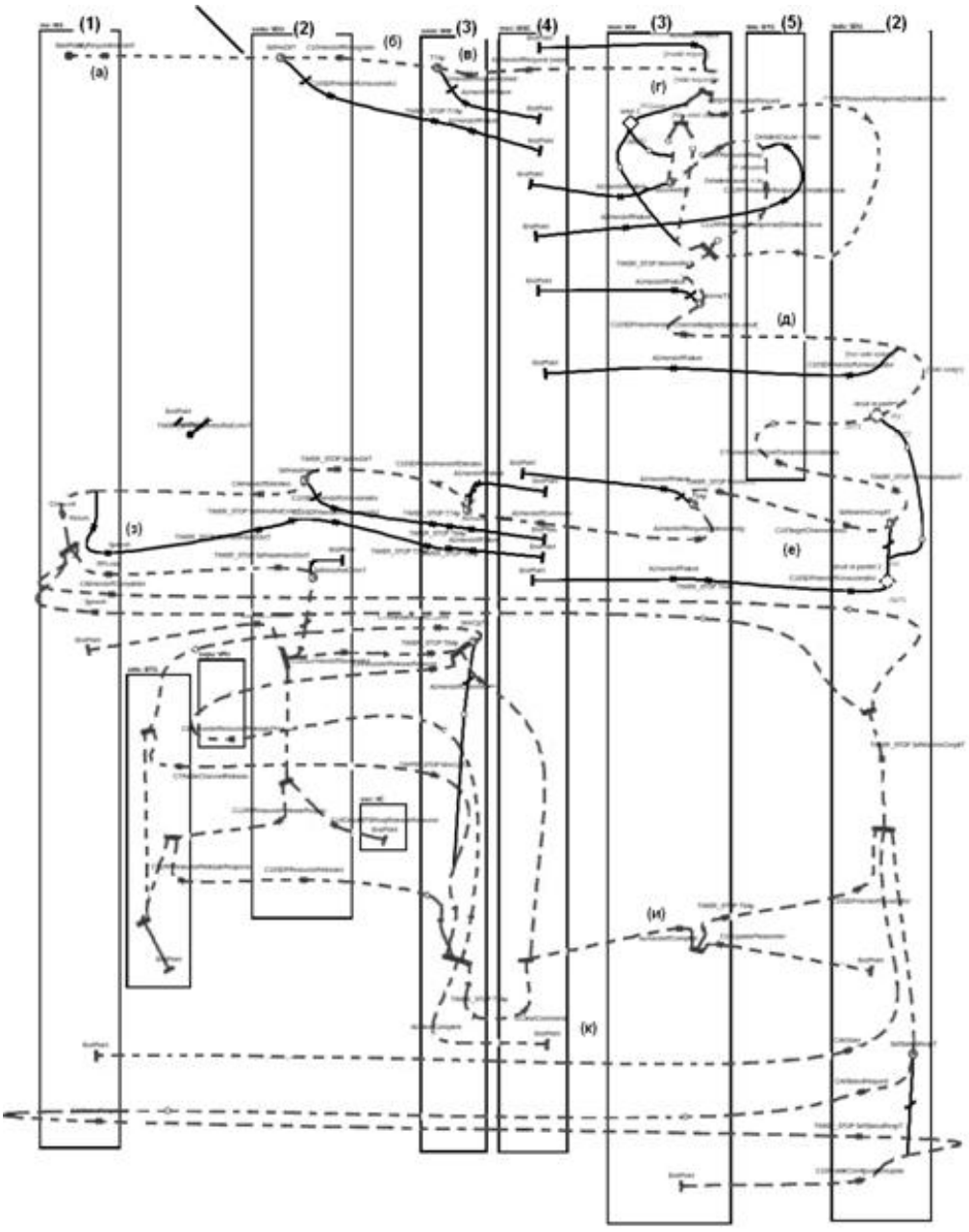

Рис. 3. Пример одного из сценариев поведения CDMA

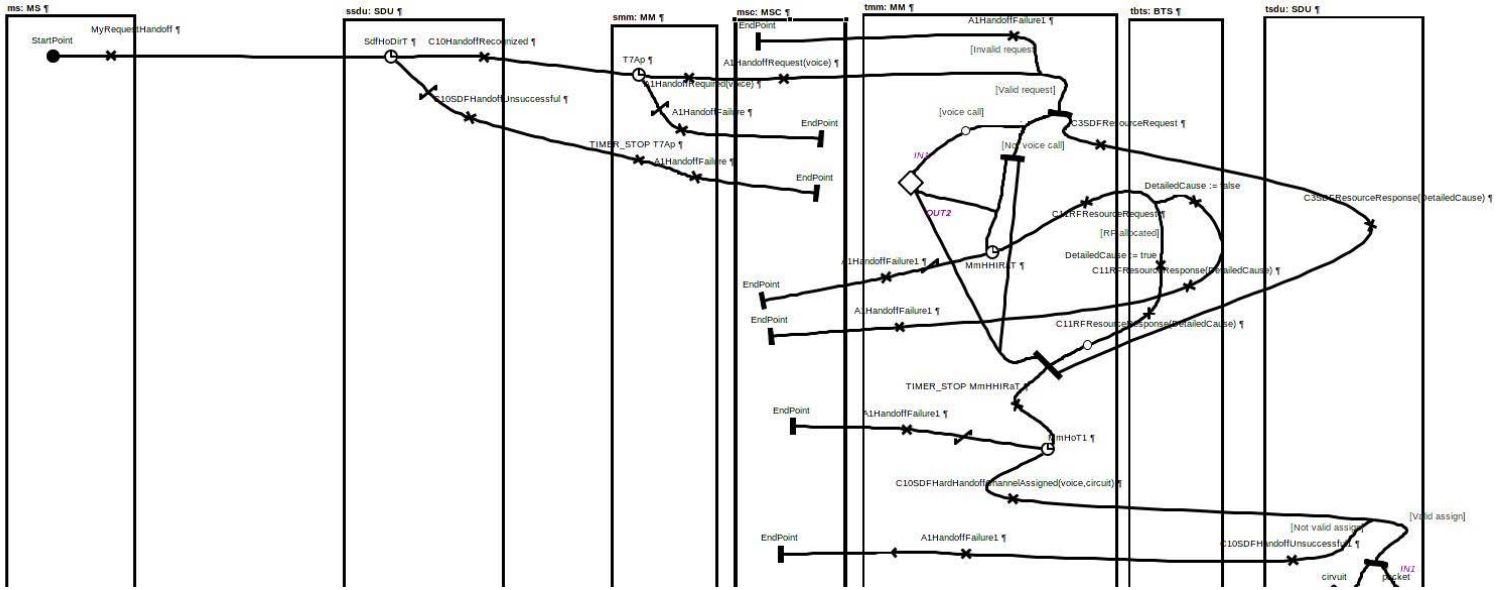

Рис. 4. Фрагмент полного сценария, описывающего подготовку к проведению процедуры handover

Рис. 5. Сокращенный сценарий за счет использования поддиаграммы Stub

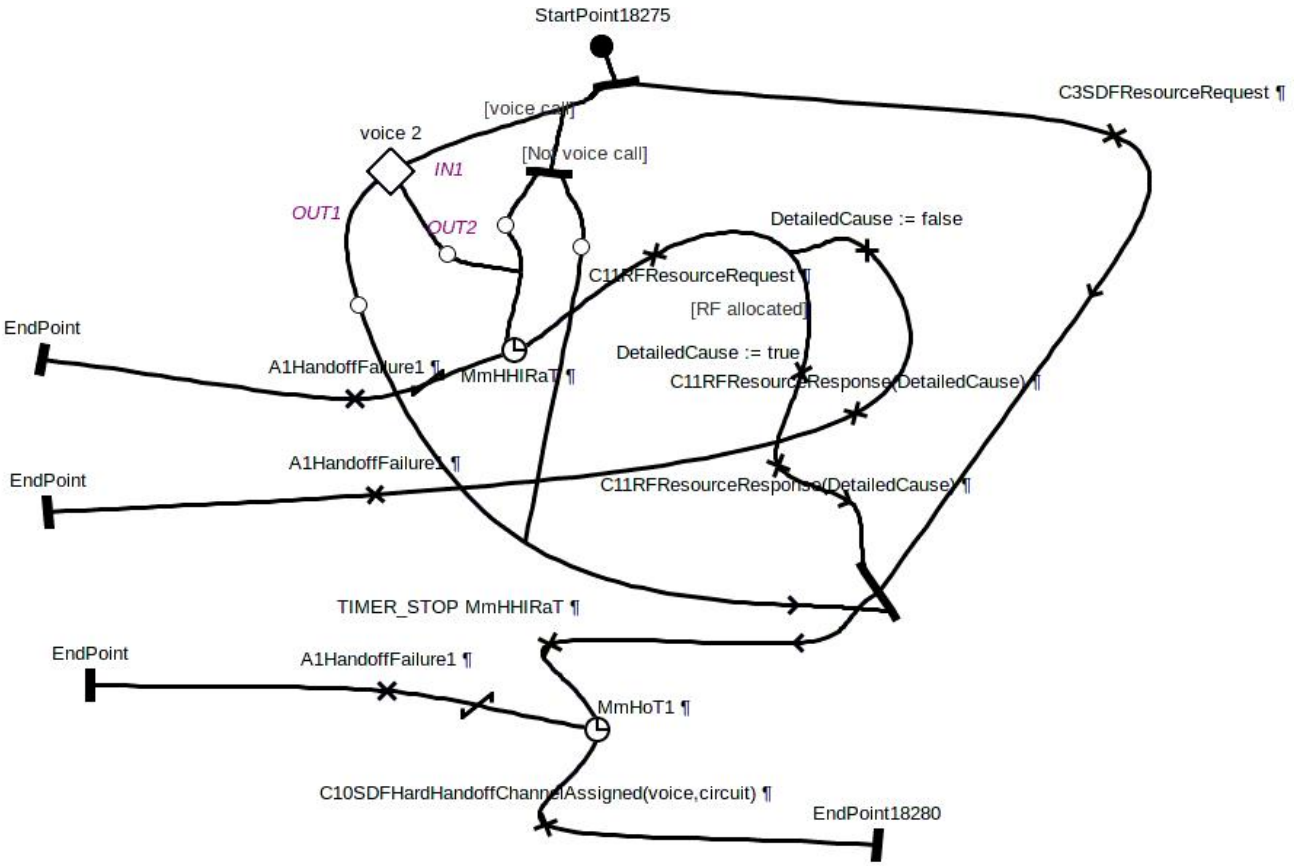

Рис. 6. Поведение, скрытое в поддиаграмме Stub

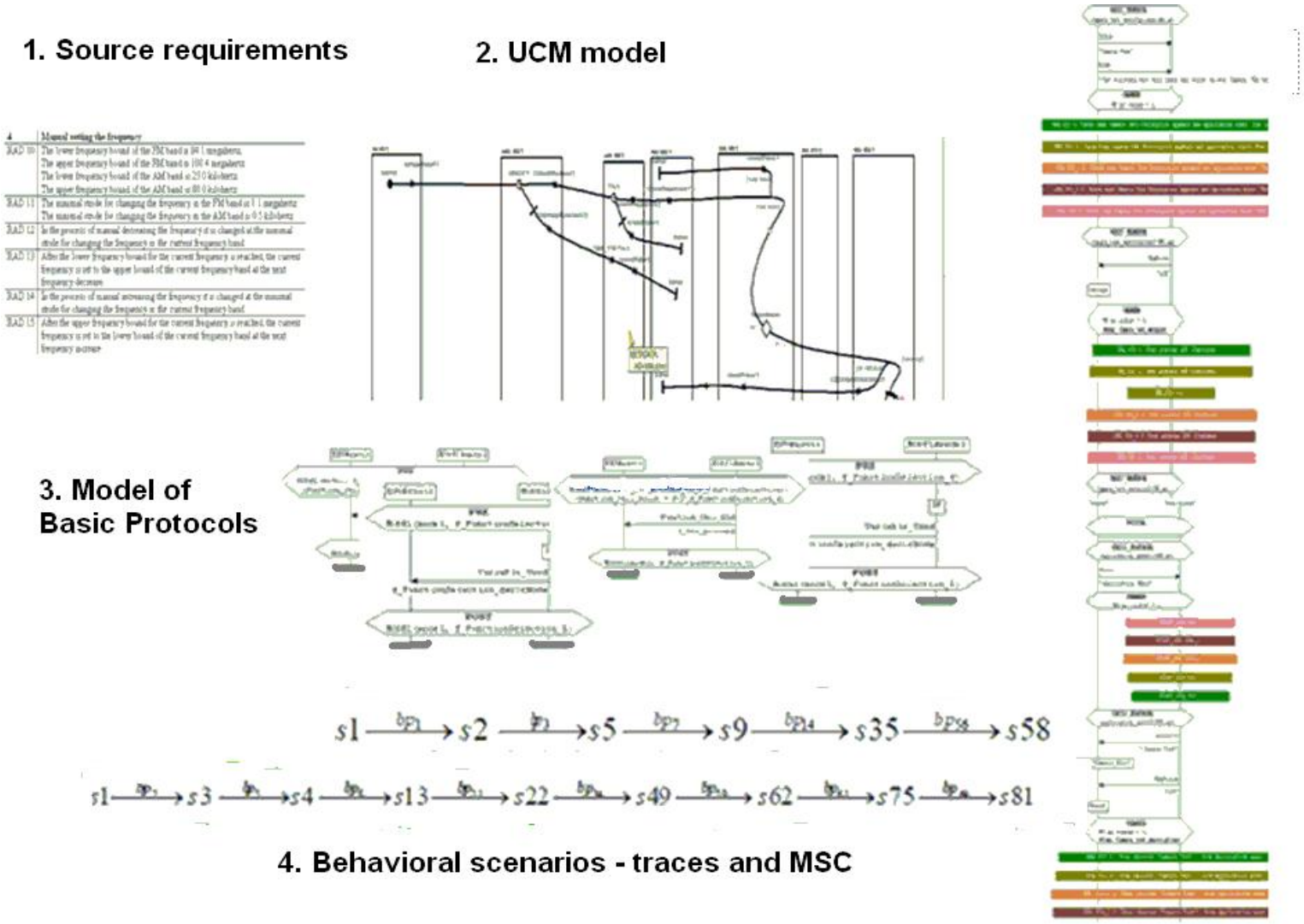

Рис. 7. Основные преобразования модели приложения

и эвристиками. Эвристики позволяют управлять процессом генерации трасс из BP формальной модели. Система автоматической верификации позволяет получить информацию о возможности появления в поведении системы недетерминизмов, дедлоков, неописанных переменных, а также определяет неполноту требований. В случаях обнаружения ошибок формальная модель или требования исправляются и вновь отправляются на верификацию. В результате получаем очищенную модель, пригодную для генерации исполнительной модели и тестов. Процесс верификации используется многократно, пока модель не достигнет достаточной степени детализации и корректности, чтобы на ее основе можно было генерировать исполняемый код.

На третьем этапе модель используется как исходное описание для генерации кода приложения на различных целевых языках, а также для генерации кода тестовых наборов по MSC-сценариям, сгенерированным по модели. Например, для генерации тестов используется специализированный кодогенерирующий шаблон, преобразующий трассу на языке MSC в целевой код теста, который может быть исполнен на целевой платформе. Покрытие тестами функциональности системы оценивается на основе полученной на этапе формализации UCM диаграммы. В случае необходимости по данной диаграмме можно сформулировать и проверить систему с учетом критериев покрытия ветвей, путей и критериальных цепочек. Другим результатом является протокол прогона теста в виде MSC диаграммы, позволяющий показать достижимость критерия покрытия требования или зафиксировать артефакт ошибки.

Применение в процессе проектирования двух моделей дает существенные преимущества, поскольку свойства моделей взаимодополняемы.

- Преимуществами UCM-модели являются:
  - ✓ близость к исходным требованиям;
  - ✓ простота и понятность при контроле и согласовании архитектурных решений с заказчиком;
  - ✓ многоуровневость описания архитектуры и поведения системы (от высокого до детального).
- Недостатки UCM-модели связаны с отсутствием инструментальных средств проверки корректности модели и генерации кода.
- Преимуществами модели базовых протоколов являются:
  - ✓ доказательство свойств модели и проверка корректности на уровне спецификаций, обеспечивающая очистку модели и пригодность для генерации исполняемого кода;
  - ✓ автоматическое создание символических тестовых сценариев, формирующих модель приложения;
  - ✓ определение областей допустимых значений модели.
- Недостатками модели базовых протоколов является трудоемкость формализации и сложность согласования проектных решений с заказчиком.

Эффективность совместного использования моделей подтверждена при проектировании телекоммуникационных проектов различной сложности.

Заключение

Настоящая работа была сфокусирована на применение нового подхода к процессу генерации моделей приложений и их тестовых сценариев на базе формализованных моделей. Особенностью подхода было применение в процессе проектирования двух формальных моделей: одной – в виде высокоуровневой нотации Use Case диаграмм, используемой для контролируемого заказчиком описания поведения и согласования с ним поведенческих сценариев, другой – в нотации базовых протоколов для последующего создания по ней тестовых сценариев. Разработанные методы были интегрированы в технологию VRS/TAT, где обеспечили 30–40 % сокращения трудоемкости проектирования телекоммуникационных приложений.

Список литературы

1. *Баранов С. Н., Котляров В. П.* Автоматизация формализации требований для получения сценариев тестирования программ // Перспективы систем информатики: Тр. семинара «Наукоемкое программирование». 15–19 июня 2009 г. Новосибирск, 2009. С. 27–35.

2. *Baranov S., Kapitonova J., Letichevsky A., Volkov V., Weigert T.* Basic Protocols, Message Sequence Charts, and Verification of Requirements Specifications // Computer Networks. 2005. Vol. 49 (5). P. 661–675.

3. *Baranov S., Kotlyarov V., Letichevsky A.* An Industrial Technology of Test Automation Based on Verified Behavioral Models of Requirement Specifications for Telecommunication Applications // IEEE Press. Proc. 8th Region. Eurocon. Conference. St. Petersburg, 2009. P. 122–129.

4. *Баранов С. Н., Котляров В. П., Летичевский А. А*. Индустриальная технология автоматизации тестирования мобильных устройств на основе верифицированных поведенческих моделей проектных спецификаций требований // Космос, астрономия и программирование. Лавровские чтения: Материалы междунар. науч. конф. СПб, 2008. С. 134–145.

5. *Kotlyarov V., Drobintsev P., Peskov D., Yusupov Y.* Implementation of an Integrated Verification and Testing Technology in Telecommunication Project // Proc. of St. Petersburg IEEE Chapter, International Conference. St. Petersburg, 2005. P. 87–92.

Материал поступил в редколлегию 19.08.2011

V. P. Kotlyarov, P. D. Drobintsev

SOFTWARE DESIGN AUTOMATION BASED ON FORMAL SPECIFICATION

Considered is the problem of manual development of specifications for the designed application, which introduces a significant efforts to the software development process together with the problem of application semantics control, required by the customer. Discussed are the instruments of development automation of requirements and architectural models' formal specifications as well as the technology of specifications and models design in the notation, appropriate for the customer's control (monitoring, inspection).

Keywords: requirements models, architectural models, semantics of the requirements, customer's control (monitoring, inspection).