САНКТ-ПЕТЕРБУРГСКИЙ ГОСУДАРСТВЕННЫЙ
УНИВЕРСИТЕТ

На правах рукописи

Соколов Владимир Владимирович

СОВМЕСТНОЕ ИСПОЛЬЗОВАНИЕ MSC И SDL МОДЕЛЕЙ ПРИ РАЗРАБОТКЕ СОБЫТИЙНО-ОРИЕНТИРОВАННЫХ СИСТЕМ

05.13.11 — Математическое и программное обеспечение
вычислительных машин, комплексов и компьютерных сетей

АВТОРЕФЕРАТ
диссертации на соискание ученой степени
кандидата физико-математических наук

Санкт-Петербург
2007

Работа выполнена на кафедре системного программирования математико-механического факультета Санкт-Петербургского государственного университета.

Научный руководитель:	доктор физико-математических наук, профессор Терехов Андрей Николаевич
Официальные оппоненты:	доктор технических наук, профессор Гольдштейн Борис Соломонович
	кандидат физико-математических наук, доцент Костин Владимир Андреевич
Ведущая организация:	Институт системного программирования РАН

Защита диссертации состоится *"26" апреля* 2007 года в 16^{30} часов на заседании диссертационного совета Д212.232.51 по защите диссертаций на соискание ученой степени доктора наук при Санкт-Петербургском государственном университете по адресу: 198504, Санкт-Петербург, Старый Петергоф, Университетский пр., д. 28, математико-механический факультет Санкт-Петербургского государственного университета.

С диссертацией можно ознакомиться в Научной библиотеке Санкт-Петербургского государственного университета по адресу: 199034, Санкт-Петербург, Университетская наб., д. 7/9.

Автореферат разослан *"21" марта* 2007 года.

Ученый секретарь
диссертационного совета
доктор физико-математических наук,
профессор Б.К.Мартыненко

Общая характеристика работы

Актуальность темы

В последнее десятилетие в развитии программного обеспечения произошел качественный скачок. Электронный мир теперь распределен, параллелен и взаимосвязан. Банки, авиалинии, телефонные компании не могут работать без интенсивного обмена информацией между всеми действующими объектами. С возрастанием сложности протоколов взаимодействия потребовались специальные средства, нацеленные на их описание, разработку и отладку.

Системы, в которых использовались сложные протоколы взаимодействия, существовали достаточно давно. Это телекоммуникационные системы. Соответствующие проблемы там были успешно разрешены, в результате чего были созданы стандарты SDL (Specification and Description Language) [15] и MSC (Message Sequence Chart) [14, 16]. Данные стандарты были разработаны Международным Консультационным Комитетом по Телеграфии и Телефонии, ныне ITU-T. MSC и SDL являются стандартом де-факто в международной телефонии. Организация ITU-T использует их для спецификации других стандартов (протоколов) — UMTS, GSM, E-DSS-1, V5.2, SS7 и других. MSC и SDL широко используются в известных фирмах – разработчиках телекоммуникационного оборудования.

MSC и SDL применяются для описания динамического поведения системы. В современных системах разработки телекоммуникационного ПО используются оба стандарта, поскольку они дают различную информацию и дополняют друг друга. MSC больше относится к этапу проектирования, а SDL — к этапу программирования. Возникает необходимость обеспечения их согласованности, поскольку они по-разному описывают одни и те же алгоритмы поведения. Параллельное использование моделей без их согласования приводит в тому, что одна модель дает корректное описание, а другая содержит устаревшую информацию. Для того, чтобы переиспользовать данные с MSC модели существует синтез SDL диаграмм по MSC [6, 20, 21]. Для проверки их согласованности существует ряд алгоритмов верификации [11, 12, 13].

Выбранные алгоритмы совместного использования моделей вносят ряд ограничений на всю технологию. Например, алгоритмы генерации могут как урезать MSC модель [19], так и запрещать модифицировать SDL модель [20], а так же влияют на согласование данных, описывающих структуру системы [20, 21] (то есть, на то, что непосредственно не относится к событийному описанию системы). Алгоритмы верификации тоже имеют свои границы применимости. Поэтому возникает проблема интеграции

различных подходов в рамках разрабатываемой технологии так, чтобы они обеспечивали как описательную, так и алгоритмическую мощь.

В общем виде задача сравнения MSC и SDL алгоритмически неразрешима, поэтому необходима разработка алгоритма, который бы абстрагировался от части информации и давал адекватные результаты. Существующие подходы плохо работают при рассогласованиях моделей, допустимых с точки зрения человеческой логики, таких как добавление и удаление сообщений, цикличности частей моделей; неполнота MSC модели и т.д.

Существенным минусом известных алгоритмов генерации является то, что в них нет возможности влиять на генерируемый SDL код. Для разработки это неудобно, поэтому автор ставит задачу о настройке SDL кода. То есть, среди ряда SDL моделей, соответствующих данному MSC, выбрать наиболее удобную для технолога[1].

Как верификация, так и генерация базируются на том, что существует MSC модель, практически полностью аналогичная SDL модели. На самом деле, существующий стандарт (H)MSC [14, 16] хорошо приспособлен лишь для описания прямых веток. То есть, хорошо приспособлен лишь для детализации сценариев поведения. Описание ошибочных ситуаций ведет к большому количеству перерисовок сценариев вместо соответствующего их дополнения. Данный момент очень неудобен для технолога. Причиной этого является то, что существующая детализация является аналогом процедуры в АЯВУ, однако логика описания ошибочных ситуаций ближе к функциям. Необходимо разработать соответствующую MSC–подобную модель.

Цели работы

Целью работы является разработка подхода по совместному использованию MSC и SDL моделей и реализация его в технологическом средстве. Разработка подхода была разделена на решение следующих основных задач:

- выработать структуру средства, предоставляющего набор возможностей совместного использования MSC и SDL с отслеживанием корректности разработки;
- разработать верификацию SDL диаграмм по MSC диаграммам при априорном знании о наличии расхождений;
- разработать алгоритм генерации SDL диаграмм по MSC диаграммам, позволяющий строить SDL код, отвечающий требованиям технолога;

[1]Так мы называем разработчика, создающего продукт с помощью технологии

– расширить MSC модель для возможности создавать максимально полные MSC описания.

Общая методика

MSC и SDL модели рассматриваются как структуры, задающие языки, состоящие из сообщений. Каждый такой язык задается с помощью конечного автомата. Разработаны алгоритмы перехода от предметной области MSC и SDL в конечный автомат и обратно. Для разбора языков использовались контекстно-свободные (LALR(1)) грамматики. Разработаны алгоритмы обработки конечного автомата на базе конечно-автоматных алгоритмов и графов, специфичные для предметной области и выбранной задачи. Оценка применимости полученных алгоритмов осуществлялась экспериментально.

Основные результаты

Предложена методология совместного использования MSC и SDL диаграмм.

Разработана математическая модель верификации SDL по MSC, позволяющая осуществлять проверки моделей в при наличии ряда отличий, появляющихся в течение жизненного цикла.

Создан алгоритм автоматизированной генерации SDL кода по конечному автомату, позволяющий получать SDL код, удобный для технолога.

Разработано расширение MSC диаграмм и соответствующий математический аппарат для работы с ним, позволяющее создавать описания реальных систем. Данное расширение может быть использовано и в генерации, и в верификации так же как и обычные MSC.

Научная новизна

Представлен оригинальный взгляд на совместное использование MSC и SDL моделей, использующий разработанные алгоритмы верификации, генерации, расширения MSC и сочетания статических данных MSC и SDL.

Разработана новая математическая модель верификации, дающая возможность проверять соответствие в тех случаях, когда другие алгоритмы не могут этого сделать.

Впервые поставлена задача настройки генерируемого SDL кода под требования технолога.

Разработан оригинальный алгоритм генерации SDL по MSC, реализующий данную настройку.

Разработано новое расширение MSC, которое позволяет создавать описания реальных систем за счет большей описательной гибкости, чем в других моделях.

Практическая и теоретическая ценность

Разработан технологический подход по совместному использованию MSC и SDL моделей, позволяющий ускорить процесс разработки и отладки ПО, существенной частью которого являются протоколы взаимодействия.

Разработана новая модель верификации, новое расширение MSC диаграмм, оригинальный алгоритм генерации SDL по MSC, используемые в данном подходе.

Разработанные подходы могут быть применены для ряда других систем, сводящихся к конечно-автоматным. Например, в подходах на базе UML [23].

Алгоритмы генерации SDL могут быть так же применены для систем, в которых SDL является промежуточной моделью [10], а так же для оптимизации существующего SDL кода.

На практике предложенные методы были использованы при разработке объектов и исследованиях кода реальных промышленных систем.

Апробация работы

Результаты работы докладывались на семинаре Института системного программирования РАН (2006 год, г. Москва).

Предложенная технология была реализована в виде набора технологических средств, дополняющих средство REAL [2, 4].

Разработанные подходы использовались при разработке телефонной станции “Юнивер–А” — многоканального радиоудлинителя телефонных линий.

Публикации

Основные результаты диссертации изложены в 3-х работах, перечисленных в конце автореферата.

Структура и объем диссертации

Диссертация состоит из введения, 4-х глав, заключения, списка литературы и 3-х приложений. Текст диссертации изложен на 146-ти страницах. Список литературы содержит 74-ре наименования.

Содержание работы

Во **введении** показывается актуальность выбранной темы исследований, излагаются научный и исторический контексты диссертационной работы. Кратко перечислены основные результаты диссертации.

Диссертация начинается с рассмотрения современных систем, показывается необходимость специальных средств для описания протоколов взаимо-

действия. Рассматриваются SDL и MSC модели, области их использования, технология REAL, на базе которой реализуются все предложенные решения. Приводится краткое введение в SDL и MSC, указывается их сходство и различие, показываются их места в процессе разработки систем. Доказывается, что нужны оба именно стандарта и необходимы специальные процедуры их согласованному использованию.

Дается обзор существующих подходов по совместному использованию MSC и SDL моделей с указанием их сильных и слабых сторон:

1. MSC и SDL независимы, что плохо из-за их несогласованности и двойного проектирования системы — сначала на MSC, потом на SDL.
2. Попытки создавать системы на основе MSC диаграмм с последующим синтезом SDL.
   (a) Только на MSC модели с использованием SDL только как промежуточной модели без возможности ее редактирования [20]. Данный подход имеет ряд недостатков, связанных с недостаточными возможностями MSC модели и с утратой выразительности SDL.
   (b) "Односторонний синтез" [6, 21], неудобный с точки зрения поддержки всего цикла разработки.
   (c) "Инкрементальные алгоритмы" [17, 19], которые требуют такого количества ограничений, что применимы лишь в очень малом классе случаев.
3. Алгоритмы сравнения используемых MSC и SDL диаграмм для проверки соответствия.
   (a) Генерация трасс[2] с MSC диаграмм с "наложением"[3] их на SDL диаграммы [11, 12].
   (b) Алгоритмы параллельного исполнения моделей с анализом внутренних состояний протокола [13].

Формулируются условия построения CASE системы: возможность независимого использования MSC и SDL моделей; генерация SDL по MSC; верификация SDL по MSC с учетом изменений, появляющихся в течение жизненного цикла системы. Описание статической части системы[4] выделяется в отдельную модель, которой пользуются и MSC, и SDL. Данная модель заменяет соответствующие возможности описания системы в каждом из стандартов. Указываются проблемные места, которые присутствуют в существующих подходах совместного использования MSC и SDL.

[2]Трассой называем цепочку сообщений

[3]Мы говорим, что трасса "наложилась", если мы представили такой вариант выполнения SDL, что при этом была порождена цепочка входящих и выходящих сообщений, специфицированная трассой

[4]Имеющиеся классы системы, сообщения, параметры сообщений, интерфейсы, порты и т.д. [15, 16]

В **первой главе** описывается математический аппарат, используемый во всей работе. Формулируются необходимые определения и теоремы, даются комментарии к ним.

Во **второй главе** рассматривается верификация SDL диаграмм по MSC диаграммам. Рассматриваются существующие подходы и их возможности, формулируется задача по верификации при наличии отличий, разрабатываются алгоритмы, решающие данную задачу. Вычисляется сложность данных алгоритмов, даются методы ее уменьшения. Приводятся примеры, иллюстрирующие подход.

Прежде всего, ставится задача по верификации SDL модели по MSC модели, когда они описывают одну и ту же систему, но при этом отличаются. Например, на MSC отсутствует описание цикличности, добавлены или убраны некоторые сообщения, произведено расщепление или слияние MSC сценариев и т.д. Описывается используемая интерпретация MSC и SDL моделей с точки зрения верификации.

Дается формальное определение алгоритма верификации и его обоснование в терминах предметной области MSC и SDL диаграмм. Сутью алгоритма являются:

1. для обрабатываемого объекта выбираются его SDL диаграммы и множество MSC диаграмм, являющееся подмножеством диаграмм, в которых он присутствует;
2. переход от MSC и SDL диаграмм к конечным автоматам, порождающих для данного объекта такой же язык из сообщений, который бы порождала соответствующая модель;
3. замена в конечных автоматах части переходов, маркированных сообщениями, на ε-переходы. То есть, устранение соответствующих переходов из рассмотрения. При автоматической работе алгоритма остаются только сообщения, присутствующие на обоих автоматах;
4. поиск такого состояния на модифицированном SDL-автомате, что начиная с него можно “наложить” все возможные трассы, задаваемые модифицированным MSC-автоматом. Количество данных трасс может быть бесконечным;
5. интерпретация результата в рамках предметной области.

Сам алгоритм дается неконструктивно, поэтому переводится в конструктивную форму на базе математики конечных автоматов. При этом задача поиска возможности вложения трасс сводится к задаче проверки включения одного языка в другой, затем сводится к задаче проверки равенства языков некоторых автоматов, что конструктивно проверяется изоморфностью их минимизированных детерминированных автоматов.

Предлагаемый подход выигрывает как у проверок на основе состояний протокола [13], так и у конечно-автоматных моделей без словарей сообщений [18] за счет интеграции нескольких идей:

- переход от SDL и MSC к конечно-автоматным моделям с учетом специфики SDL и MSC (конструкции параллельного исполнения MSC, уничтожение и сохранение сообщений, не обрабатываемых в текущем SDL-состоянии и т.д.);
- операции фильтрации, разделяющие общее множество сообщений на "используемые" и "неиспользуемые";
- проверки на включение языков, реализуемой с помощью математики конечных автоматов.

Это дает возможность проверять соответствие MSC и SDL диаграмм на соответствие в ряде случаев, когда другие алгоритмы не применимы.

В **третьей главе** рассматривается генерация SDL диаграмм по MSC диаграммам. Дается обзор существующих алгоритмов, обсуждаются их возможности, описывается предлагаемый подход. Для этого показываются принципы построения отдельных элементов SDL, после чего выводится алгоритм построения всего кода. Предлагаются методы вмешательства в данный алгоритм для улучшения генерируемого SDL.

Проводится обзор существующих алгоритмов по генерации SDL из MSC. Фактически существует две группы — серия московских работ [3, 20] и серия канадских [6, 17, 21]. Во всех данных методах сначала осуществляется переход от MSC к конечному автомату. Наиболее существенное отличие между ними в том, что в канадских работах генерация начинается сразу, а в московских к конечному автомату перед этим применяются алгоритмы детерминизации и минимизации. Канадские алгоритмы выдают код, в котором лучше угадывается изначальный MSC, более точно рассчитывают правила для конструкции Save, но из-за отсутствия алгоритма детерминизации применимы не всегда, а из-за отсутствия минимизации могут выдавать менее оптимальный код. Московский вариант может осуществить синтез всегда, но при этом он может как улучшать компактность SDL, так и ухудшать ее, а так же понижать степень узнаваемости MSC в SDL.

Отдается предпочтение связке детерминизация – минимизация для повышения области применимости, но следует выполнить ряд действий для того, чтобы убрать негативные влияния данных алгоритмов на полученный SDL. В работе приводится соответствующий пример.

Основной подход работы заключается в том, что конечная SDL модель предназначается для человека, поэтому неуправляемого алгоритма синтеза мало, технолог должен иметь возможность получить тот вариант SDL кода,

который лучше удовлетворяет его требованиям, среди ряда возможных, реализующих одну и ту же задачу.

Для этого вводится промежуточная модель между MSC и SDL диаграммами для настройки процедуры синтеза. Это конечный автомат. Он более компактен, чем MSC и SDL модели, и удобен для редактирования перед последующей генерацией SDL. На результат генерации можно повлиять используя данные из модели MSC, модели SDL и на уровне конечных автоматов. Например, два параллельно выполняющихся MSC сценария, при выполнении ряда ограничений, удобно реализовывать тремя участками SDL кода, которые выполняют каждый из сценариев и процедуру, которая создает и завершает исполнение данных "сценариев", а так же передает сообщения на них. SDL модель мы используем когда работаем с уже готовым SDL кодом — например, используем описатели "*" для состояний и сообщений, а так же описатель "–" для состояний при выделении общих участков кода. Иногда это может существенно уменьшить его объем.

Наиболее существенной, с точки зрения влияния на получаемый SDL, является область работы с конечными автоматами. При этом конечный автомат представляется в виде графа, с которым удобно работать визуально. На данном графе изображены состояния конечного автомата и переходы между ними. Переходы несут информацию о имени сообщения и его типе — посылаемое или принимаемое. С конечным автоматом можно работать либо и до, и после связки процедур детерминизация–минимизация для защиты от изменений, либо после выполнения данных алгоритмов. Работа "и до, и после" путем замены части сообщений на отсутствующие в списке, а потом замена обратно решает проблему детерминизации во выходящим сообщениям, иногда негативно влияющую на SDL код. Работа "и до, и после" путем замены целой области и замены обратно позволяет защитить ее от изменений, вносимых данными алгоритмами. Работа "после" позволяет "переразложить" код нужным образом, а так же интерактивно выделять процедуры путем обнаружения гамаков [1].

При построении SDL диаграмм по MSC диаграммам для выбранного объекта проходим по следующей цепочке:

1. проверка корректности MSC описания [6, 8];
2. переход от MSC диаграмм данного объекта к недетерминированному конечному автомату, порождающему все трассы данного объекта [3];
3. опциональная защита частей автомата от процедур преобразования;
4. детерминизация автомата;
5. минимизация автомата;
6. "восстановление" автомата, если применялись алгоритмы пункта 3;

7. опциональное “переразложение” автомата и выделение процедур;
8. построение схемы SDL кода.

Данный алгоритм позволяет настраивать генерируемый SDL код под требования технолога, чего нет ни в одном из алгоритмов генерации. Поскольку при генерации SDL диаграмм по MSC диаграммам используется переход MSC→FSM→SDL, то вместо MSC диаграмм может быть задана любая модель, которая сводится к конечным автоматам. Это открывает возможность переноса информации из других технологий в SDL. Особо следует отметить, что исходной моделью может также служить и SDL модель. Таким образом можем проводить оптимизацию уже написанного SDL кода.

В **четвертой главе** вводится предлагаемое расширение MSC. Для этого сначала показываются важные недостатки существующего стандарта MSC, затем описывается предлагаемое решение на базе разработанного математического аппарата. Предложенный подход обсуждается и иллюстрируется примерами.

Для результативной работы алгоритмов генерации и верификации необходимо, чтобы существовало MSC описание системы, которое было бы как можно более полным. Существующий стандарт MSC имеет серьезный недостаток, не позволяющий описывать в нем обратные ветви (то есть, ветви, обрабатывающие ошибочные ситуации). Он хорошо работает для описания прямых ветвей (случаев, когда все происходит без ошибок) путем постепенной декомпозиции сценариев. Когда на набор сценариев, описывающих прямые ветви, начинают добавлять проработку ошибочных ситуаций, то оказывается, что изменение детализирующего сценария влияет на детализируемый. Зачастую данное изменение на этом не заканчивается, а продвигается “выше” по иерархии сценариев. В реальной работе это приводит к достаточно большому количеству “перерисовок” сценариев, на что тратится время.

Причина данной проблемы достаточно очевидна — из сценария невозможно вернуть результат его выполнения с последующим анализом в детализируемом сценарии. При этом важным граничным условием является то, что решение данной проблемы не должно быть сложнее конечно-автоматной модели для последующего использования в генерации и верификации. Соответствующая модель была разработана; идеологически она является смесью графического MSC и текстовых описаний; при необходимости графические и текстовые описания могут возвращать результат выполнения; данный результат выполнения может быть проанализирован в текстовом описании с выполнением последующих действий примерно так же, как это делается на языке Pascal; полное описание на данной модели конверти-

руется в конечный автомат для каждого из объектов для последующего использования.

```
function f2;
begin
    while f1=2 do
        if f3>1 then
            return 1
        else
            proc4
        fi;
    return 2;
end

procedure main;
begin
    if f2=1 then
        proc5
    fi;
end
```

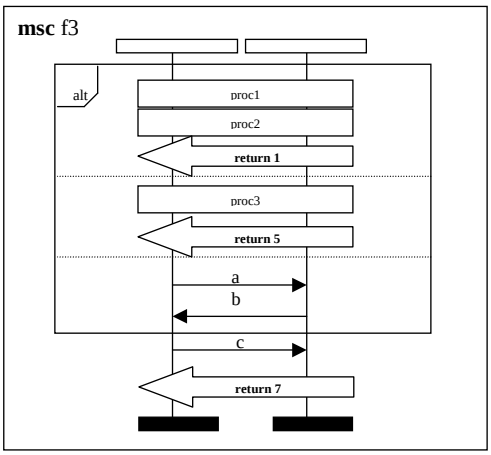

Рис. 1: Расширение графических диаграмм для добавления возможности возвращать результат

Слева от рис. 1 приведен образец текстового описания предлагаемого расширения MSC. На самом рисунке изображено необязательное расширение графических сценариев возможностью возвращения результата его выполнения. Как графические, так и текстовые описания делятся на “процедуры” и “функции”. “Процедуры” не возвращают результат. “Функции” результат возвращают. На примере продемонстрировано совместное использование текстовых и графических описаний, и организация “функций” на базе “процедур” и обратно. По “процедуре” для любого объекта может быть построен конечный автомат, описывающий его язык, состоящий из входящих и выходящих сообщений.

Если проводить аналогии с обычными сценариями, то “процедура” является сценарием, а “функция” является набором нумерованных сценариев. Можно считать, что сценарии в “функции” являются вариантами выполнения аналогично конструкции *alt*, и номер присвоен концу сценария. При использовании “функций” в текстовом описании данный набор можно разделить необходимым образом согласно их нумерации и соединить с нужными последующими действиями. Для текстовых описаний реализованы конструкции: *if; case; for; while; opt; alt* с Pascal’е-подобным синтаксисом.

При разборе “программы” на предложенном расширении MSC для фиксированного объекта используются структуры, изображенные на рис. 2. В графическом изображении данной структуры вершина S обозначает начало блока (аналогия начального состояния конечного автомата). Вершина K обозначает конец блока, при последовательном соединении блоков K со-

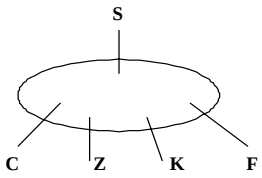

Рис. 2: Структура, используемая при разборе расширения MSC (блок)

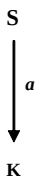

Рис. 3: Блок, соответствующий посылке одного сообщения

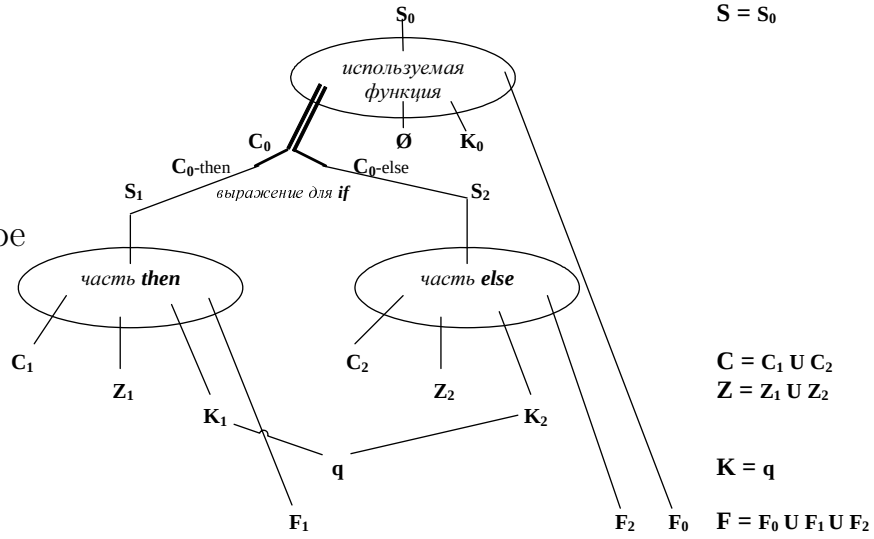

Рис. 4: Использование блоков при разборе оператора *if* общего вида

единяется с S; на рис. 3 приведен блок для одного сообщения. Множество вершин F обозначает завершение исполнения (аналогия множества завершающих состояний конечного автомата). Z — множество вершин выхода из "процедуры" (*return* без кода возврата). C — множество вершин выхода из "функции", маркированных кодом возврата (например, *return* 7).

Внутри блока находятся конечно-автоматные структуры, описывающие прием и посылку сообщений. В процессе разбора грамматики текстового расширения MSC происходят операции с данными блоками. На них же определены проверки корректности "программы". Графические представления так же отображаются в подобные блоки. Блок, получающийся в конце разбора, является "процедурой" и отображается в конечный автомат.

На рис. 4 показано использование блоков при разборе оператора *if* общего вида. Множество кодов возврата "функции" (C_0) на основе выражения разбивается на 2 подмножества (C_{0-then} и C_{0-else}). К соответствующим вершинам одного подмножества присоединяется блок ветви *then*, а к другому — *else* через начальные вершины данных блоков. Концы блоков ветвей (K_1 и K_2) соединяются с дополнительной вершиной q. В результате из трех блоков ("функция", блок ветви *then* и блок ветви *else*) получаем один блок, соответствующий оператору *if*. В качестве S для него будет выступать S_0, в качестве C объединение C_1 и C_2, в качестве Z — Z_1 и Z_2, в качестве K — q, в качестве F — объединение F_0, F_1 и F_2.

Рассмотрим место предложенного решения среди диаграмм, которые позволяют описать алгоритмы взаимодействия нескольких объектов, и могут

давать все поведение группы объектов или системы, а не разрозненных информационных срезов. Достаточно большое количество моделей, аналогично MSC, рассчитаны лишь на постепенную детализацию без проработки обратных ветвей, и имеют похожие проблемы. Это и MSC-подобная группа — HMSC [14]; UML Sequence [23] и UML Collaboration [23] диаграммы; Somé's Scenarios [24]. Это и попытка представить поведение системы в виде конечного автомата Chisel Diagrams [7]. Это и потоки данных — UML Activity диаграммы [23] и Use Case Maps (UCMs) [5]. Предлагаемое решение превосходит их по гибкости, поскольку логика стыковки различных вариантов поведения лучше описывается текстом. Так же следует упомянуть текстовые языки, нацеленные на описание взаимодействия различных объектов (Pascal–FC, Occam, Ada, Java, Lotos и т.д.), но они предназначены для описания поведения одного объекта, и лежат вне рассматриваемой области.

Задача по расширению MSC для описания обратных веток уже ставилась — это LSCs (Life Sequence Charts) [9]. Предлагаемое решение перекрывает LSCs за счет возможности иметь произвольное множество кодов возврата, а не двух вариантов — нормального и ошибочного. Так же предлагаемое решение в рамках всего подхода сводимо к широко используемому SDL, а не требует отдельной модели для исполнения, как LSCs.
Существует модель CTP [22], являющаяся комбинацией MSC и сетей Петри. Разработанное решение лучше за счет комбинирования АЯВУ и MSC. Выигрыш получается за счет того, что технологам при разработке протоколов привычно думать в терминах циклов и условных операторов, нежели в терминах сетей Петри.

В **заключении** приведена краткая характеристика технологического подхода, разработке которого было посвящено данное исследование, и сформулированы основные полученные результаты.

В **Приложении А.** приведены используемые элементы MSC и SDL диаграмм. В **Приложении В.** дается грамматика предложенного расширения MSC. **Приложение С.** содержит список иллюстраций.

Список литературы

[1] Касьянов В.Н. Оптимизирующие преобразования программ. — М.:Наука, 1988. — 336 с.

[2] Кознов Д.В. Визуальное моделирование компонентного программного обеспечения: Дис... канд. физ.-мат. наук — СПбГУ, 2000. — 82 с.

[3] Мансуров Н. Синтез исполняемых SDL спецификаций по сценарным моделям // Тр. ин-та Системного Программирования — 1999.

[4] Терехов А.Н. и др. REAL: методология и CASE средство разработки информационных систем и ПО систем реального времени // Программирование. — 1999. — № 5. — с. 45-51.

[5] Use Case Maps (UCMs) нотация. Определения, публикации, средства. http://www.useCaseMaps.org/index.shtml

[6] Abdalla M., Khendek F., Butler G. New Results on Deriving SDL Specifications from MSCs // Proc. of 9th SDL Forum — 1999. — P. 55-67.

[7] Aho A., Gallagher S., Griffeth N., Scheel C., Swayne D. Sculptor with Chisel: Requirements Engineering for Communications Services // Proc. in 5th International Workshop on Feature Interactions in Telecommunications and Software Systems (FIW'98) — Lund, Sweden: IOS Press, 1998. — P. 45-63.

[8] Ben-Abdallah H., Leue S. Syntactic detection of process divergence and nonlocal choice in message sequence charts // Proc. of the Tools and Algorithms for the Construction and Analysis of Systems, Third International Workshop (TACAS'97) — № 1217 in Lecture Notes in Computer Science — Enschede, The Netherlands: Springer, 1997. — P. 259-274.

[9] Damm W., Harel D. LSCs: Breathing Life into Message Sequence Charts // Proc. in 3rd IFIP International Conference on Formal Methods for Open Object-based Distributed Systems (FMOODS'99) — Kluwer Academic Publishers, 1999. — P. 293-312.

[10] Daveau J.-M., Marchioro G., Valderrama C., Jerraya A. VHDL generation from SDL specifications // Proc. of the XIII IFIP Conf. on Computer Hardware Description Languages (CHDL'97) — Toledo, Spain — 1997.

[11] EG 201 015 Ver. 1.2.1, Methods for Testing and Specification (MTS); Specification of protocols and services; Validation methodology for standards using Specification and Description Language (SDL); Handbook — 1999. — 28 p.

[12] ETR 184 : Methods for Testing and Specification (MTS); Overview of validation techniques for European Telecommunication Standards (ETSs) containing SDL — 1995. — 27 p.

[13] Holzmann G. Design and Validation of Computer Protocols // Prentice-Hall, 1991 — ISBN 0-13-539834-7. — 512 p.

[14] ITU-T MSC2000R3 Draft Z.120: Message Sequence Charts ITU-T Recommendation Z.120. — 1999.

[15] ITU-T Recommendation Z.100: Specification and description language (SDL). — 1999. — 244 p.

[16] ITU-T Recommendation Z.120: Message Sequence Chart (MSC). — 1999. — 136 p.

[17] Khendek F., Vincent D. Enriching SDL Specifications with MSCs // Proc. of 2nd Workshop of the SDL Forum Society on SDL and MSC (SAM2000) — 2000. — P. 305-319.

[18] Kurshan R. Computer-Aided Verification of Coordinating Processes // Princeton Series in Computer Science, 1994. — ISBN 0691034362. — 270 p.

[19] Li J., Horgan J. Applying formal description techniques to software architectural design // Computer Communications — Vol. 23 — № 12. — 2000. — P. 1169-1178.

[20] Mansurov N., Zhukov D. Automatic synthesis of SDL models in Use Case Methodology // Proc. of 9th SDL Forum — 1999. — P. 225-240.

[21] Robert G., Khendek F., Grogono P. Deriving an SDL specification with a given architecture from a set of MSCs // Proc. of 8th SDL Forum — 1997. — P. 197-212.

[22] Roychoudhury A., Thiagarajan P.S. Communicating Transaction Processes // Proc. in IEEE International Conference on Application of Concurrency in System Design (ACSD'2003) — 2003. — P. 157-166.

[23] Rumbaugh J., Jacobson I., Booch G. The Unified Modeling Language Reference Manual — Addison-Wesley, 1999. — 576 p.

[24] Somé S., Dssouli R., Vaucher J. Toward an Automation of Requirements Engineering using Scenarios //Journal of Computing and Information 2(1) — 1996. — P. 1110-1132.

Работы автора по теме диссертации

1. Соколов В.В. Проверка SDL диаграмм по MSC диаграммам на основе частичной информации // Системное программирование. — СПб.: 2004. — с. 366-390.
2. Терехов А.Н., Соколов В.В. Новые возможности технологии REAL // Вестн. С.-Петербург. ун-та. — Сер. 10., 2005. — Вып. 1-2. — с. 64-77.
3. Терехов А.Н., Соколов В.В. Реализация стыка между MSC– и SDL–диаграммами в технологии REAL // Программирование. — 2007. — № 1. — с. 35-50.
   Terekhov A., Sokolov V. Implementation of the Conformation of MSC and SDL Diagrams in the REAL Technology // Programming and Computer Software (Engl., Transl.) — 2007. — № 1. — P. 24-33. — DOI: 10.1134/S0361768807010045.