УДК 004.4'22+004.415.2+519.687.4+004.434 Вестник СПбГУ. Сер. 10, 2005, вып. 1

А. Н. Терехов, В. В. Соколов

НОВЫЕ ВОЗМОЖНОСТИ ТЕХНОЛОГИИ REAL

1. Введение. Коллектив кафедры системного программирования и ГП «Терком» занимается разработкой программного обеспечения (ПО) для телефонных станций и других систем реального времени с начала 1980-х годов. Оказалось, что самым трудным моментом является взаимодействие заказчиков, алгоритмистов, программистов, специалистов по протоколам, электронщиков и т. д. Причем проблемы возникают не из-за недостаточной квалификации кого-либо из участников разработки, а из-за взаимного недопонимания. Для преодоления этих трудностей Международный консультационный Комитет по телеграфии и телефонии (МККТТ, ныне ITU-T) разработал серию графических способов описания, в частности языки SDL [1] и MSC [2].

SDL (Specification and Description Language) диаграмма очень похожа на традиционные блок-схемы, но с несколькими важными расширениями: символ состояния, в котором процесс не занимает процессор, ожидая приема одного или нескольких сигналов; символ приема сигнала и символ посылки сигнала (например, см. рис. 9). В SDL диаграммах основное внимание уделяется логике использования сигналов; логика выполнения действий, не связанных с сигналами, детализируется лишь по мере необходимости. Поэтому действие может быть как атомарным, так и достаточно сложным из десятков или сотен операторов.

MSC*) (Message Sequence Chart) диаграммы позволяют описывать сценарии поведения системы во времени. Время течет сверху вниз, вертикальные линии представляют объекты системы, а между ними рисуются стрелки, обозначающие сигналы**). (Пример простой MSC диаграммы можно посмотреть на рис. 8). MSC диаграммы состоят из отдельных сценариев и структур, задающих последовательности выполнения этих сценариев. Они широко применяются для описания протоколов различными международными организациями, в том числе для стандартов, разработанных организацией ITU-T, однако редко когда приводятся исчерпывающие описания поведения системы, включая обработку аварийных ситуаций, техобслуживания, тарификации и т. д. И MSC, и SDL диаграммы используются для описания динамического***) поведения системы, их сравнительный анализ приведен в таблице.

В современных системах используются оба стандарта, поскольку они дают различную информацию о динамическом поведении объекта и дополняют друг друга. Соответственно возникает необходимость обеспечения их согласованности.

В данной статье опишем применение MSC и SDL диаграмм в нашей технологии REAL [5], подходы по их согласованию, в частности способы автоматической генерации SDL диаграмм по набору сценариев в виде MSC диаграмм.

2. Первые подходы. При разработке программного средства мы решили не изобретать велосипед и воспользоваться рекомендациями МККТТ. Первый редактор SDL

© А. Н. Терехов, В. В. Соколов, 2005

*)Сейчас все большее распространение получают HMSC диаграммы [3], которые являются развитием MSC диаграмм. По своей сути они во многом остаются такими же, поэтому отдельно они не рассматриваются, и термином MSC обозначаем и их тоже.

**)В данной статье SDL понятие сигнала считаем эквивалентным MSC понятию сообщения.

***)Эти модели имеют средства описания и статической части системы, но проблемы в этой области были решены в рамках нашей технологии ранее [4].

Таблица 1. **Сравнительные характеристики SDL и MSC**

MSC	SDL
Описывает взаимодействие между различными частями системы; по отношению к объекту является внешним	Описывает внутренность каждого объекта; по отношению к объекту является внутренним
Отображена логика взаимодействия нескольких объектов. Логика поведения каждого из них скрыта	Отображена логика поведения каждого из объектов. Логика их взаимодействия скрыта
Информация о выборе линии поведения дается словесно	Логика работы детализована вплоть до значений переменных
Создаются на этапе проектирования	Создаются при программировании
Используются при обсуждении задачи с экспертами предметной области	Преимущественно используются программистами

диаграмм был реализован на ЕС ЭВМ в 1984 г. Конечно, графическим его можно было назвать лишь с некоторой натяжкой, поскольку и экраны, и устройства печати были только текстовыми. Однако, если забыть про некоторую громоздкость SDL диаграмм, они выглядели как настоящие.

Первые опыты по применению SDL редактора на реальных телефонных станциях оказались неудачными – представьте себе две-три сотни больших печатных листов диаграмм, набранных с помощью SDL редактора и напечатанных на АЦПУ, которые Вы можете только читать и водить пальцем по связям. Поэтому мы разработали конвертор SDL диаграмм в код высокого уровня, систему имитационного моделирования, различные отладчики, средства для снятия и анализа трасс, другие инструменты, которые обеспечили первые практические применения SDL диаграмм.

В начале 1990-х годов все технологические средства перевели на ПЭВМ, а заодно начали использовать объектно-ориентированный подход к проектированию ПО. Был разработан язык, являющийся в каком-то смысле текстовым аналогом диаграммы классов, с помощью которого описывались объекты, их интерфейсы, внутренние атрибуты, и т. д. (в настоящее время есть очень похожая разработка OMG – IDL [6]). Удалось найти эффективную реализацию вычислительного процесса для систем реального времени; усилить различные статические проверки, например, если сигнал не упомянут в описании объекта, то его невозможно использовать в SDL диаграммах; разработать средства автоматической генерации данных и т. д.

Под именем RTST (Real Time Software Technology) [7] эта технология просуществовала около 10 лет, с ее помощью были разработаны 8 типов различных телефонных станций и несколько других программно-аппаратных средств, причем SDL диаграммы легко переносились с одной платформы на другую. Существенно было облегчено сопровождение ПО, введение в коллектив новых специалистов, переиспользование фрагментов, короче, в настоящее время мы являемся убежденными сторонниками графических средств проектирования.

При дальнейшем развитии данной технологии она была переименована в REAL. На момент написания статьи [5] в ней существовал разрыв между MSC и SDL диаграммами (рис.). Он являлся существенным минусом технологии, поэтому был разработан ряд средств по его преодолению. В настоящей статье покажем мировые подходы к заполнению разрыва и укажем место разработанных нами подходов среди них.

3. Существующие решения по заполнению разрыва. Первоначальные ре-

Рис. 1. Место SDL и MSC моделей в создании ПО на момент написания статьи [5].

шения предлагали MSC диаграммы только как спецификацию, а исполняемый код строился по SDL моделям. У данного подхода есть несколько очевидных минусов:

1) динамика обмена сообщениями создается дважды – сначала при проектировании, а затем при программировании. При повторном создании информация не переиспользуется;

2) несогласованность MSC и SDL моделей при изменениях.

Первая проблема породила попытку автоматизированной генерации SDL диаграмм по MSC диаграммам, а вторая – необходимость проверки соответствия между ними.

Сначала рассмотрим возможность автоматической генерации. На текущий момент авторы данной статьи выделяют следующие основные модели синтеза SDL диаграмм по MSC диаграммам:

1. Группа работ, выполненных в канадском университете Конкордиа (Concordia University, Canada) [8–11]. В дальнейшем эту модель синтеза будем называть «канадской».

2. Работа [12], выполненная в Институте системного программирования Российской Академии наук, на основе которой было создано средство MOST [13], авторы которого принимали участие в создании средства KLOCWork [14, 15]. Эту модель будем называть «российской».

Для того чтобы показать сильные и слабые стороны возможности синтеза, произведем сравнение двух данных алгоритмов между собой по следующей шкале:

- статическая часть SDL модели;
- схема построения динамического поведения;
- характеристики построенной динамической части SDL модели, по возможности их связь с алгоритмом построения.

Для алгоритмов работы «канадской» модели характерно:

1. Сначала фиксируется SDL архитектура (разбиение на объекты, определение каналов связи между объектами, перечней сообщений каждого канала). Архитектура может быть задана средствами SDL модели [8] либо сгенерирована из UML [11]. После этого автоматическая генерация SDL диаграмм по MSC диаграммам осуществляется в уже зафиксированную архитектуру.

2. Схема построения динамического поведения:*)

(a) анализ возможности построения;

(b) составление таблиц частичного упорядочивания сигналов с учетом архитектуры. При этом для каждого из каналов строится отношение предшествования между сигналами либо говорится, что сигналы не находятся в подобном отношении;

(c) выделение конечных автоматов из MSC;

(d) построение SDL кода по автоматам с учетом таблиц частичного упорядочивания сигналов, которые используются для вычисления необходимого перечня сохраняемых сигналов конструкции Save.

3. В процессе синтеза динамической части SDL диаграмм вносимые изменения минимальны. Следствием этого являются:

- построенная SDL модель обладает «наглядностью», это означает, что в полученной SDL модели легко узнается исходная MSC модель;
- использование конструкции Save с точным указанием сохраняемых сигналов;
- излишне громоздкая SDL модель даже там, где ее можно было бы оптимизировать;
- для некоторых MSC моделей невозможно произвести синтез таким образом, хотя выполняющая ту же логику SDL модель существует и может быть построена другими алгоритмами синтеза (отсутствие детерминизации автомата).

Для алгоритмов «российской» модели свойственно:

1. По MSC модели синтезируется не только динамическое поведение, но и архитектура SDL модели.

2. Схема построения динамического поведения:

(a) выделение конечных автоматов из MSC;

(b) детерминизация полученных автоматов;

(c) минимизация автоматов;

(d) построение SDL кода по автоматам.

3. При синтезе динамического поведения возможно автоматическое его преобразование. Как следствие:

- Динамика SDL модели может быть изменена относительно исходной MSC модели. Из-за этого SDL модель становится менее узнаваемой (процедуры детерминизации и минимизации).
- Более грубое использование конструкции Save, подразумевающее сохранение всех сигналов (трудности с «протаскиванием» точной информации через алгоритмы детерминизации и минимизации).
- Возможность синтеза даже в тех случаях, когда предыдущий алгоритм выдает невозможность реализации (использование детерминизации).

*) Алгоритмы статей приводятся в слегка модифицированном виде для облегчения их сравнения.

- Возможность построения более простых (оптимальных) SDL моделей в некоторых случаях, по сравнению со спроектированными MSC диаграмами (применение минимизации).
- Возможное ухудшение SDL модели в некоторых случаях (детерминизация и по выходным сигналам, хотя это и не является необходимым для возможности сгенерировать SDL модель из конечного автомата).

Теперь перейдем к необходимости согласования MSC и SDL моделей. Для начала рассмотрим проблему с точки зрения алгоритмов генерации, поскольку одной из мотиваций автоматической генерации и была сложность сравнивания этих моделей.

1. В алгоритмах «канадской» модели предлагается возможность изменения исходных MSC диаграмм с соответствующим инкрементальным изменением SDL диаграмм [10]. Это интересный подход, однако авторы [10] упоминают, что он будет работать не во всех случаях.

2. В алгоритмах «российской» модели предлагается вообще не заниматься проверкой соответствия MSC и SDL диаграмм и запрещено ручное развитие сгенерированных SDL диаграмм. После каждого внесенного на MSC диаграммы изменения необходимо заново сгенерировать SDL модель. Для этого в данных алгоритмах предлагается расширить MSC модель операциями с переменными, параметрами, условными состояниями, чтобы при генерации получать не шаблон SDL диаграмм, а готовую модель. У этого подхода много недостатков, поскольку MSC диаграммы больше отвечают на вопрос «Что может случиться?», чем на вопрос «Почему так случается?», вследствие чего они обладают избыточностью и мало приспособлены для использования на них «арифметических» данных.

Существуют алгоритмы согласования MSC и SDL диаграмм, не связанные с автоматической генерацией:

1) алгоритм верификации на основе трасс*) без пропускания сигналов, когда по MSC диаграммам строятся наборы трасс, затем их пытаются «накладывать» на SDL диаграммы без пропускания сигналов. Если трасса укладывается, то все хорошо. Если нет – это считается ошибочной ситуацией;

2) идея модификации предыдущего алгоритма с разрешением пропускать сигналы [16]. Однако она распространения не получила из-за невысокой достоверности результата;

3) использование алгоритмов, связанных с анализом внутренних состояний протокола [17], при этом параллельно выполняются две модели – MSC и SDL. В процессе выполнения проверяется, что принятые и посланные сигналы одинаковы.

Более детальное описание алгоритмов 1 и 3 можно найти в стандартах серии ETSI [18, 19], их реализация имеется в средстве Telelogic Tau [20].

4. Недостатки существующих подходов, их анализ и предложенные решения. Работы по согласованию MSC и SDL диаграмм продолжаются, поэтому следует сформулировать ряд проблем, которые плохо решаются существующими средствами, и изложить свои соображения по их решению.

4.1. ***Необходимость использования модифицированных MSC диаграмм.*** Первое, что хочется отметить, так это то, что авторы существующих алгоритмов по генерации SDL диаграмм подразумевают создание MSC диаграммы, которая бы полностью описывала все поведение объекта. После чего можно будет применить процедуру синтеза и получить SDL код различной степени готовности. На самом деле это

*)В данной статье под трассой понимаем цепочки сообщений.

очень сильное утверждение, и в реальности создание исчерпывающих описаний очень трудоемко. Виной этого недостаточная выразительная способность стандарта MSC. Он разрешает выполнять функциональную декомпозицию примерно так же, как проводится декомпозиция статики,– разрешает разбивать блоки на более мелкие части, а те дробить на еще более мелкие.

У такого подхода есть существенный недостаток. В реальных задачах по декомпозиции функциональной модели выполнение блока подразумевает наличие результата, который можно проанализировать и в дальнейшем выбрать один из множества путей развития. В существующем стандарте MSC такой возможности нет, поэтому хорошо описываются лишь прямые ветви алгоритмов, а описание исключительных ситуаций дается с большим трудом.

Для того чтобы иметь возможность использовать алгоритмы генерации, был создан метаязык (расширение языка MSC), который позволяет:

1) вернуть результат выполнения сценария;

2) проанализировать его и выбрать необходимый вариант поведения.

Это дало нам возможность описывать структуры, подобные следующим:

```
function сценарий_1(Обьект1, Объект2, Объект3) {
    if(сценарий_2(Обьект1, Объект2, Объект3)!=1){
        сценарий_3(Объект2,Обьект3);
        сценарий_4(Объект2,Обьект3);
    } else {
        while(сценарий_5(Обьект1, Объект2)==7) {
            if(!сценарий_6(Обьект1, Объект2))
                return 0;
        }
    }
    return 1;
}
```

С помощью подобных моделей в разработанном нами технологическом средстве эффективно описываются обратные ветви. Наш подход позволяет это делать более гибко, чем в наиболее близкой системе [21].

4.2. ***Алгоритмы синтеза.*** Сразу отметим, что невозможность осуществить синтез во многих разумных ситуациях является очень слабым местом, поэтому сразу стоит отказаться от «канадской» модели в пользу использования «российского» алгоритма. Далее будем обсуждать именно его с точки зрения возможных улучшений.

Авторы «российского» алгоритма расширили стандарт MSC для поддержки всего цикла разработки программ и запретили редактирование SDL диаграмм. С нашей точки зрения, это было не самое удачное решение, и от него следует отказаться в пользу разрешения редактирования SDL. Тем самым обеспечивается больший комфорт на стадии программирования за счет использования SDL модели, лучше приспособленной для программирования. За это приходится платить необходимостью согласования моделей, и возникает процедура верификации.

И MSC, и SDL модели в принципе позволяют описывать структуру системы, но, поскольку архитектура у системы едина, вся работа со статикой была вынесена в отдельный редактор на основе диаграммы классов, а все остальные диаграммы заимствуют информацию из созданных в нем моделей. Так мы избавились от согласования статик этих моделей.

С алгоритмом построения динамики сложнее. Цепочка детерминизация – минимизация обладает как рядом плюсов, так и рядом минусов, упомянутых выше. Еще раз

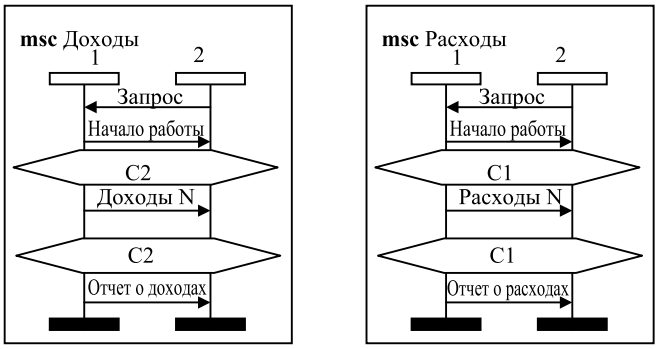

Рис. 3. Детализирующие MSC диаграммы для генерации.

поясним, что детерминизация необходима для того, чтобы расширить область применимости алгоритма, а минимизация необходима для нейтрализации побочных эффектов детерминизации, когда автомат мог получиться слишком большим.

Применяемый нами подход базируется на идее, что конечная SDL модель предназначается для человека, поэтому неуправляемого алгоритма синтеза мало, проектировщик должен иметь возможность получить тот или иной SDL код по своему вкусу. Это означает, что некоторые участки кода могут быть «неправильными»*), но удобными для последующей доработки.

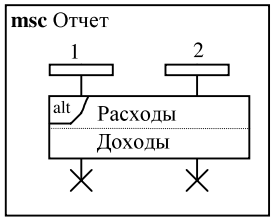

Рис. 2. Главная MSC диаграмма для генерации.

Поэтому была введена промежуточная модель между MSC и SDL диаграммами для настройки процедуры синтеза. Это конечный автомат. С одной стороны, он уже несет информацию только о поведении данного объекта и игнорирует поведение других объектов, а с другой – он еще не нагружен специальной информацией о логике принятия решений и ограничениями на конечно-автоматную модель SDL. Из-за этого он более компактен, чем MSC и SDL модели, и удобен для редактирования перед последующей генерацией SDL. В качестве примера можно рассмотреть рис. 2–5. Автомат редактируется человеком перед и после связки процедур детерминизация – минимизация. Этим мы обеспечиваем:

- генерацию SDL кода в наиболее удобном виде;
- возможность защиты от процедур оптимизации;
- исключение эффекта минимизации по выходным сигналам;
- выделение SDL процедур в конечном автомате;
- возможность синтеза SDL кода из параллельных сценариев интеллектуальным способом.

В заключение приведем пример, связанный с возможным ухудшением SDL модели при использовании «российского» алгоритма. На рис. 2, 3 показаны исходные MSC диаграммы, на рис. 6 – SDL модель, которая получена из «российского» алгоритма, на рис. 5 – модель, полученная с помощью предложенного подхода. Она была получена с помощью детерминизации модели только по входным сигналам. Автомат, из

*)Например, быть недетерминированными по выходным сигналам либо быть неоптимальными с точки зрения процедуры минимизации.

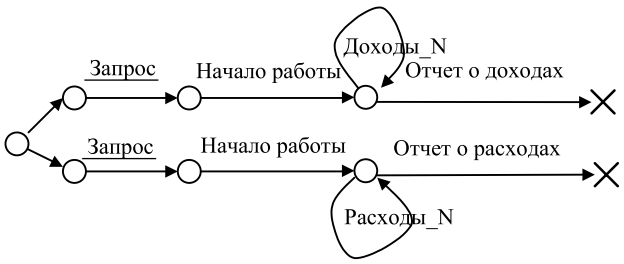

Рис. 4. Автомат, полученный из MSC диаграмм 2 и 3.

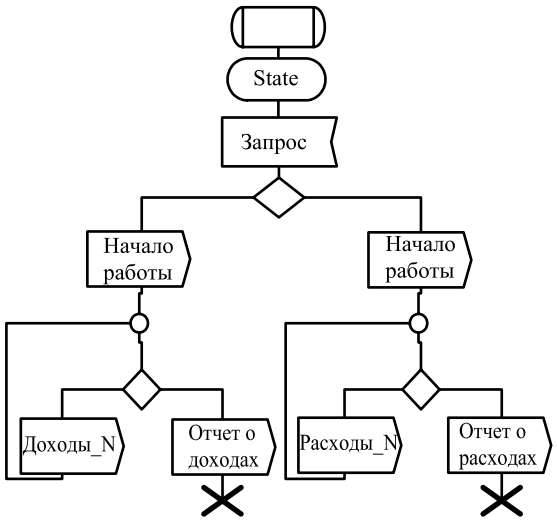

Рис. 5. Наш результат генерации.

которого она была построена, не является детерминированным из-за наличия сигнала «Начало работы». Но этот сигнал является выходным, а значит, подобная конструкция допустима. Видно, что она проще и по логике лучше соответствует исходным спецификациям.

Отметим, что наш алгоритм синтеза может работать ровно точно так же, как и «российский» без какого-либо вмешательства. При этом их результаты будут совпадать. А вот если понадобится более корректное решение, то можно будет перейти в режим ручного редактирования и получить более «человечные» SDL тексты.

4.3. ***Алгоритмы проверки соответствия MSC и SDL моделей.*** Обычно используются такие подходы:

1. Сравниваются два белых ящика. Проверяется, совпадают или нет их входные и выходные последовательности. Если совпадают, то системы считаются одинаковыми; если нет, то находится последовательность, где начинается расхождение. Этот подход можно назвать «глобальным».

2. Другой подход – предъявляется трасса и проверяется, может ли она быть выполнена начиная с некоторого SDL состояния. Он обеспечивает только локальную проверку.

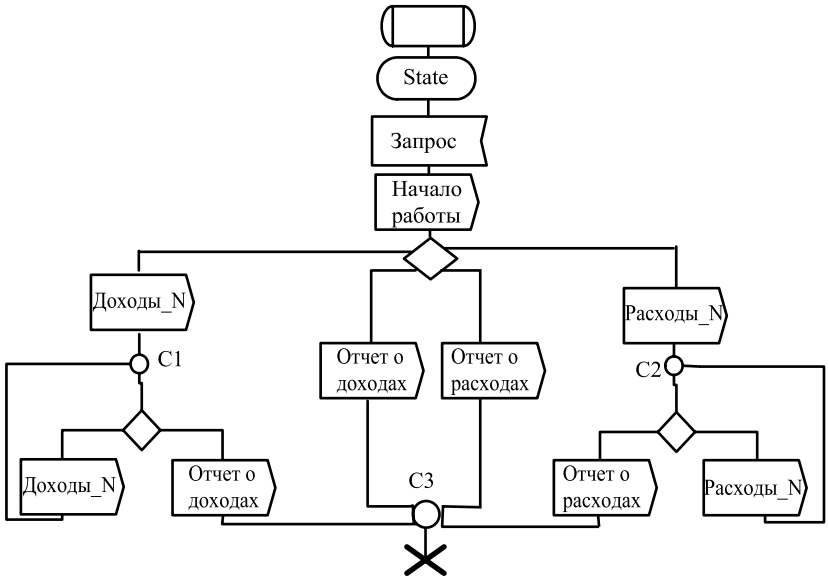

Рис. 6. «Классический» результат генерации.

Фактически рассматриваются две крайности. Часто MSC и SDL модели похожи, особенно, если последняя была получена из первой. Но в реальных разработках MSC модель не доводится до полного соответствия с SDL моделью, она и остается только похожей! Процедура верификации должна уметь выдавать результат и в таких случаях.

Рассмотрим несколько примеров:

1) на MSC диаграммах спроектированы две различные роли, их реализациям соответствует один SDL объект;

2) MSC диаграммы отражают поведение лишь одного объекта, а при реализации он оказался разбитым на два;

3) код из SDL автомата был модифицирован вставкой дополнительных сигналов, в частности для накопления отладочной информации;

4) на MSC диаграммах, например, был спроектирован лишь обмен между двумя из трех сущностей. Обмен с третьей был посчитан более простым и не документировался. А на SDL, естественно, был реализован обмен между всеми тремя объектами;

5) MSC диаграммы показывают лишь основные варианты развития системы без описания стыковки вариантов между собой. А в SDL коде все эти ветви были собраны в единый код с возможной вставкой сигналов с одних вариантов в другие. С точки зрения человека, подобная реализация подобна документации.

В Telelogic Tau для проверки соответствия можно воспользоваться одним из следующих подходов*):

1) из MSC диаграммы сделать трассу, которую потом наложить на SDL диаграммы;

2) использовать какой-либо алгоритм параллельного выполнения MSC и SDL моделей для проверки соответствия состояний либо нахождения различий в них. (Алгоритмы подробно описаны в [17].)

*) Даются основные идеи, для более детальных подходов см. документацию на Telelogic Tau [20].

При этом используются следующие условия на порядок сообщений на MSC и SDL диаграммах:

- трасса должна содержать все сообщения, которые присутствуют на MSC;
- трасса не должна содержать других сообщений;
- порядок сообщений должен быть одним и тем же.

В противном случае это считается ошибкой.

Из всех рассмотренных трудностей Telelogic Tau реально может справиться лишь с первыми двумя примерами за счет параллельного выполнения двух моделей. При наличии какой-нибудь другой проблемы Telelogic Tau будет выдавать ошибку верификации MSC. Это означает, что Telelogic Tau может выполнить либо глобальную проверку, либо локальную. Но на вопрос, соответствует ли конкретная SDL модель множеству MSC диаграмм, ответ будет получен не всегда.

Для проверки соответствия в реальных задачах был разработан собственный алгоритм [23]. Он базируется на таких подходах:

1. И MSC, и SDL диаграммы переводятся в конечные автоматы, порождающие все возможные цепочки сообщений данной модели.

2. Над автоматами проводятся специальные модификации с учетом предметной области: это учитывание конструкции Save; уничтожение сигналов, не обрабатываемых в текущем SDL состоянии; сохранение MSC цепочек, которые не ведут к уничтожению объекта, и др.

3. Данные конечные автоматы проходят операцию «фильтрации» по наборам сообщений. Сообщения, не прошедшие фильтрацию, заменяются на ε-переход в конечном автомате. При этом выкидываются сообщения, которые не принадлежат словарю сообщений обеих моделей.

4. Проверка включения языков, заданными конечными автоматами [22]. SDL-автомат проверяется на наличие такой вершины, что все трассы любой длины MSC-автомата укладываются, начиная именно с нее.

Этим алгоритмом мы реализуем проверку соответствия со следующими свойствами.

1. Философия проверки базируется на определении из [24] о минимальном требуемом поведении. Идейно оно означает, что MSC диаграммы являются априори верными, а SDL диаграммы – проверяемыми. При этом «любая»*) последовательность сигналов с MSC диаграмм должна укладываться на SDL диаграммы. Это означает, что SDL реализация должна быть способна обработать все трассы, специфицированные на MSC.

2. Проверяем, что у SDL реализации есть хотя бы одна возможность правильно обработать MSC-спецификацию. Это означает, что если у системы есть два варианта выполнения, один из которых ведет к ошибке, а другой – к положительному результату, то выбираем «положительный». Например, когда сигнал нужен и может прийти вовремя, а может прийти заранее и уничтожится из-за несохранения в состоянии, то выбирается случай, когда он придет вовремя.

3. Ошибкой считается ситуация, когда существует трасса с MSC спецификаций, не имеющая отображения в SDL реализацию. Как бы мы корректно не исполняли программу, все равно она не соответствует спецификациям**).

4. Возможно, что программа будет обрабатывать еще какие-либо трассы или из-за

*) С точностью до использования словарей сообщений – фильтрации.

**) Для случаев, когда поведение SDL программы зависит от деталей реализации (например, использование конструкции Save, уничтожение сигналов, не обрабатываемых в состоянии), алгоритм проверки может быть «настроен».

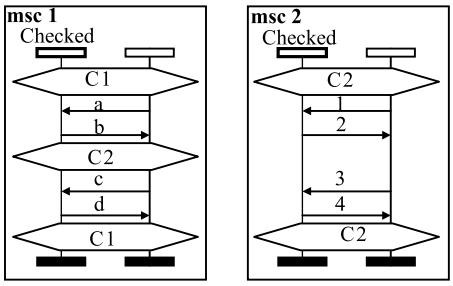

Рис. 7. MSC диаграммы 1, 2 для проверки соответствия.

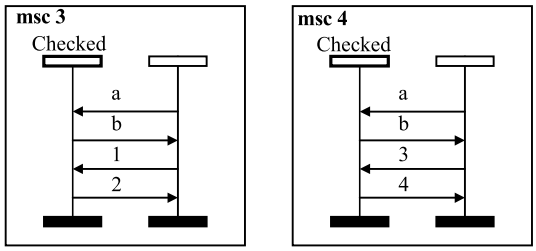

Рис. 8. MSC диаграммы 3, 4 для проверки соответствия.

ошибок в программе будет выполнять лишь часть всех вариантов поведения. В данном подходе рассматриваем только принципиальную возможность выполнения. Это означает, что мы не опускаемся до анализа переменных, поскольку тогда задача становится алгоритмически неразрешимой, а рассматриваем только возможные цепочки сообщений в конечном автомате, соответствующем SDL коду.

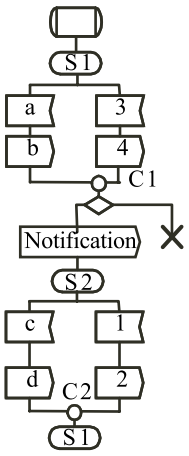

Рис. 9. Проверяемая SDL диаграмма.

Приведем пример. Пусть существуют MSC спецификации 1, 2, 3, 4. Спецификации 1 и 2 изображены на рис. 7, а спецификации 3 и 4 – на рис. 8. Еще дана SDL реализация, показанная на рис. 9. Ставится целью проанализировать соответствие реализации спецификациям. Наш метод позволяет выявить, что SDL реализация:

1) удовлетворяет MSC спецификации 1 без учета спецификаций 2, 3 и 4;

2) удовлетворяет MSC спецификации 2 без учета спецификаций 1, 3 и 4;

3) удовлетворяет спецификациям 1 и 2 вместе, если начальной является спецификация 1*);

4) удовлетворяет спецификациям 1 и 2 вместе, если начальной является спецификация 2;

5) удовлетворяет спецификации 3;

6) не удовлетворяет спецификации 4.

*) В случае с HMSC диаграммами момент начала работы объекта указывается однозначно, а в случае с MSC диаграммами это надо указать специальным образом – выбрать одну из MSC диаграмм в качестве начальной.

5. Заключение. Предложенные подходы по стыковке MSC и SDL диаграмм графически объединены на рис. 10. Выделим основные идеи, использующиеся в нашей технологии, и результаты их применения.

1. Единый редактор для структур данных, гарантирующий согласованность статических частей моделей.

2. Предложенное расширение MSC диаграмм, которое лучше позволяет описывать обратные ветки, чем обеспечивается возможность создания более полных MSC моделей для использования их при генерации и верификации.

3. Функции генерации, которые по MSC модели позволяют полуавтоматически получить скелет SDL кода и этим существенно сократить время разработки.

4. Возможность наращивания SDL фрагментами на языке высокого уровня для последующей генерации исполняемого кода. Это дает возможность использования SDL модели не только на этапе проектирования, но и на этапе программирования без дополнительных переходов.

5. Возможность проверки соответствия SDL диаграмм MSC документации при изменении любой из данных моделей. Таким образом мы контролируем отсутствие противоречий на данных моделях.

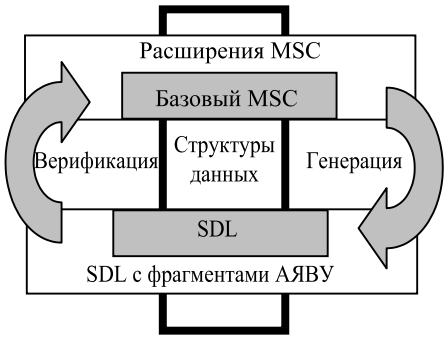

Рис. 10. Объединенный подход к согласованию MSC и SDL в нашей технологии.

Предложенная технология позволяет ускорить процесс создания ПО и повысить качество получаемого продукта за счет максимальной состыковки всех моделей технологии (см. также [5, 4]). Модели выбраны таким образом, что они достаточно хорошо покрывают область телекоммуникационных систем и систем реального времени.

Рассмотренные в данной статье алгоритмы могут быть применены и в ряде других технологий, использующих вместо MSC модели другую, сводящуюся к конечно-автоматной. В частности, подобные алгоритмы могут быть применены в технологиях на базе моделей [6]. Так же они применимы в ряде других подходов, в которых SDL модель является промежуточной [25].

Авторы предвидят дальнейшее развитие метода проверки непротиворечивости SDL модели MSC документации для решения следующих задач в верификации, которые на текущий момент не имеют удовлетворительного решения:

1) нахождение того, что может выполнять SDL модель кроме того, что указано в MSC документации;

2) определять покрытие SDL реализации MSC документацией.

Summary

Terekhov A. N, Sokolov V. V. New trends of Real technology.

The MSC and SDL diagrams jointing are considered with the aim of automating manual conversion from system behaviour scenarios (HMSC/MSC) to behaviour models of all objects (SDL). The conformity support is discussed as coordinated changes making in a whole life cycle. The existing algorithms overview is suggested, their weak and strong fields are examined. The new approaches to overcome existing algorithms drawbacks are suggested and integrated into the whole complex.

Литература

1. ITU-T Recommendation Z.100: Specification and description language (SDL), 11/99.

2. ITU-T Recommendation Z.120: Message Sequence Chart (MSC), 11/99.

3. ITU-T MSC2000R3 Draft Z.120: Message Sequence Charts, 11/99.

4. *Кознов Д. В.* Визуальное моделирование компонентного программного обеспечения: Канд. дисс. СПб., 2000.

5. *Терехов А. Н.* и др. Real: методология и CASE средство разработки информационных систем и ПО систем реального времени // Программирование. 1999. № 5. С. 44–52.

6. Сервер группы OMG, занимающейся развитием методологии UML. Стандарты, публикации, средства // http://www.omg.com.

7. *Терехов А. Н.* RTST – технология программирования встроенных систем реального времени // Зап. семинара кафедры системного программирования «CASE-средства RTST++». СПб., 1998. Вып. 1, С. 3–17.

8. *Robert G., Khendek F., Grogono P.* Deriving an SDL specification with a given architecture from a set of MSCs // Proc. of the 8th SDL Forum/ Eds: A. Cavalli, A. Sarma. Evry, France, 1997. P. 197–212.

9. *Abdalla M. M.,Khendek F., Butler G.* New results on deriving SDL Specifications from MSCs //Proc. of 9th SDL Forum / Eds: R. Dssouli, G. V. Bochmann, Y. Lahav. Montreal, Canada, 1999. P. 51–66.

10. *Khendek F., Vincent D.* Enriching SDL specifications with MSCs // Proc. of the 2nd Workshop of the SDL Forum Society on SDL and MSC (SAM2000). Grenoble, France, 2000. P. 305–319.

11. *Bourduas S., Khendek F., Vincent D.* From MSC and UML to SDL // Proc. of IEEE Annual Intern. Conference on Computer Software and Applications (COMPSAC'2002). Oxford, UK, 2002. P. 153–158.

12. *Mansurov N., Zhukov D.* Automatic synthesis of SDL models in Use Case Methodology // Proc. of 9th SDL Forum / Eds: R. Dssouli, G. V. Bochmann, Y. Lahav. Montreal, Canada, 1999. P. 225–240.

13. *Проекты* Института системного программирования Российской Академии наук // http://www.ispras.ru/groups/case/projects.html?2#2

14. *Сайт* компании, представляющей средство KLOCWork // http://www.klocwork.com/.

15. Аннотация средства KLOCWork на сайте SDL Forum // http://www.sdl-forum.org/Tools/klocwork.htm.

16. *Sinclair D., Stone B., Clynch G.* An object oriented methodology from requirements to validation // Proc. of the 2nd Object Oriented Information Systems Conference (OOIS'95), Dublin, Ireland, 1995. P. 265–286.

17. *Holzman G. J.* Design and validation of computer protocols. Prentice-Hall, 1991. 512 p.

18. ETR 184: Methods for Testing and Specification (MTS). Overview of validation techniques for European Telecommunication Standards (ETSs) containing SDL. 1995.

19. 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.

20. *Документация* на средство Telelogic Tau 4.2. March, 2001.

21. *Damm W., Harel D.* LCSs: Breathing life into message sequence charts // Proc. 3rd IFIP Intern. Conference on Formal Methods for Open Object-based Distributed Systems (FMOODS'99). Florence, Italy, 1999. P. 293–312.

22. *Vardi M., Wolper P.* An automata-theoretic approach to automatic program verification // Proc. of the 1st Annual IEEE Symposium on logic in computer science (LICS'1986). Boston, MA, 1986. P. 332–344.

23. *Соколов В. В.* Проверка SDL диаграмм по MSC диаграммам на основе частичной информации // Системное программирование. СПб., 2004. С. 366–390.

24. ITU-T Recommendation Z.500, Framework on formal methods in conformance testing, 05/97.

25. *Muth A.* SDL-based design of application specific hardware for Hard Real-Time systems. München, 2002. 208 p.

Статья поступила в редакцию 21 апреля 2005 г.