УДК 004, 004.05, 004.4
DOI: 10.17223/19988605/53/12

В.А. Непомнящий, Е.В. Бодин, С.О. Веретнов

АНАЛИЗ И ВЕРИФИКАЦИЯ SDL-СПЕЦИФИКАЦИЙ РАСПРЕДЕЛЕННЫХ СИСТЕМ С ПОМОЩЬЮ ЯЗЫКА DYNAMIC-REAL

Описан новый программный комплекс SRDSVer3 (SDL/REAL Distributed Systems Verifier), предназначенный для моделирования, анализа и верификации SDL-спецификаций распределенных систем. Он включает транслятор из языка SDL в разработанный ранее язык Dynamic-REAL (dREAL), системы автоматического моделирования dREAL-спецификаций и их верификации с помощью системы SPIN. Этот комплекс применяется для верификации системы управления сетью банкоматов.
Ключевые слова: верификация распределенных систем; анализ распределенных систем; трансляция; SDL; SPIN.

В последние годы заметно возрастает роль формальных методов, применяемых для разработки распределенных систем, таких как системы управления и коммуникационные протоколы. Для разработки и спецификации распределенных систем часто используется язык SDL [1–3]. Поэтому проблема анализа и верификации SDL-спецификаций является актуальной.

Эта проблема рассматривалась в работе [4], где описано применение системы SPIN для верификации SDL-спецификаций с использованием промежуточного языка IF [5]. В отличие от этого подхода, в нашем подходе наряду с системой верификации SPIN применяется система моделирования поведения SDL-спецификаций, а также система поиска ошибок в SDL-спецификациях с помощью полученных контрпримеров. Это позволяет проводить анализ SDL-спецификаций и тогда, когда система SPIN неприменима.

В данной работе описывается новый программный комплекс SRDSVer3 (SDL/REAL Distributed Systems Verifier), предназначенный для моделирования, анализа и верификации SDL-спецификаций. Данный комплекс использует промежуточный язык спецификаций Dynamic-REAL (dREAL) [6].

Входной язык этого комплекса включает базовые конструкции языка SDL-2010, такие как BLOCK, PROCESS, SIGNALROUTE, CHANNEL, SAVE. Для описания свойств SDL-спецификаций используется язык SPLan (SDL Properties Language) [Там же], основанный на логике линейного времени (LTL), расширенной кванторами по экземплярам процессов. Для моделирования и анализа SDL-спецификаций также используется язык запросов. Для верификации SDL-спецификаций используется транслятор из языка dREAL во входной язык Promela системы верификации SPIN.

Описанный подход применяется для верификации и анализа динамической системы управления сетью банкоматов. Были рассмотрены две версии такой системы: корректная версия и версия, содержащая логическую ошибку. С помощью этого подхода данная ошибка была обнаружена.

Заметим, что в работе [7] описано применение системы SPIN для верификации системы управления сетью банкоматов, заданной на языке Promela. Однако система управления сетью банкоматов, описанная в разд. 5, является более мощной, а ее спецификация на языке SDL более точно описывает поведение сетей банкоматов.

1. Программный комплекс для моделирования, анализа и верификации SDL-спецификаций

Программный комплекс SRDSVer3 предназначен для моделирования, анализа и верификации спецификаций, представленных на языке SDL, с помощью промежуточного языка dREAL.

Комплекс SRDSVer3 (рис. 1) состоит из следующих компонентов:

– Транслятор1 переводит SDL-спецификацию в dREAL-спецификацию.

– Конвертер2 переводит запросы для SDL-спецификаций в запросы для dREAL-спецификаций.

– Система моделирования проводит анализ dREAL-спецификаций с помощью пользовательских запросов. Если запрос выполняется, конфигурация системы и результаты передаются в Конвертер4. В противном случае пользователь получает сообщение о невыполнении запроса.

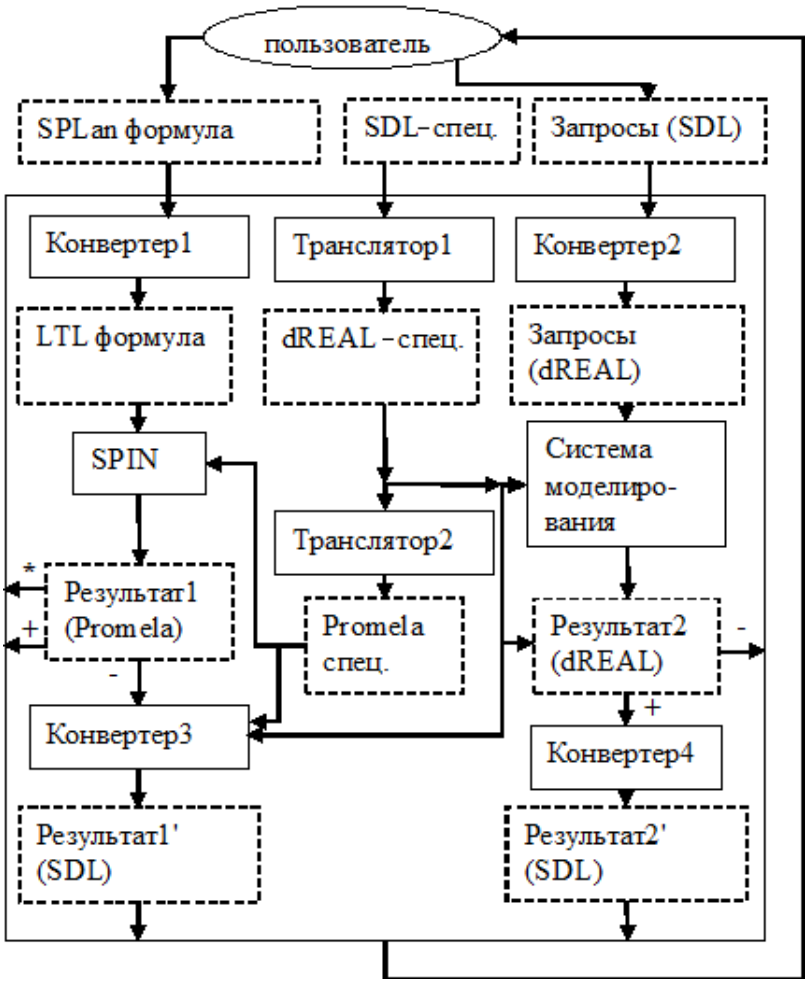

Рис. 1 Программный комплекс SRDSVer3
Fig. 1. Software package SRDSVer3

– Конвертер4 переводит Результат2 работы Системы моделирования из терминов dREAL в термины SDL. На выходе пользователь получает номер строки исходной SDL-спецификации, где выполняются условия запроса, и конфигурацию моделируемой системы в момент выполнения этих условий.

– Транслятор2 переводит dREAL-спецификацию в Promela-спецификацию.

– Конвертер1 переводит SPLan-формулу для SDL-спецификации в LTL-формулу.

– Система верификации SPIN проверяет Promela-спецификацию, полученную от Транслятора2, с учетом LTL-формулы, полученной от Конвертера1.

– Конвертер3 переводит Результат1 верификации системой SPIN в Результат1’ в терминах SDL.

Поэтому пользователь получает номер строки в исходной SDL-спецификации, где свойство, заданное LTL-формулой, не выполняется. Если это свойство выполняется, то пользователь получает соответствующее сообщение. Если при верификации Promela-спецификации не хватило памяти или размера массивов, пользователь также получает соответствующее сообщение (отмечено на рисунке звездочкой).

2. Транслятор из SDL в dREAL

Результатом работы этого транслятора является dREAL-спецификация, эквивалентная входной SDL-спецификации. Блоки, процессы, типы данных и переменные языка SDL имеют прямые аналоги в языке dREAL. В языке dREAL процессы соединены каналами, причем каждый канал хранит проходящие по нему сигналы независимо от других каналов, в отличие от общего входного буфера процесса в языке SDL. Каналы языка dREAL однонаправленные и не могут разветвляться, в отличие от каналов и маршрутов языка SDL. Операторы DECISION, EXPORT, IMPORT и динамические конструкции преобразуются в соответствующие конструкции языка dREAL. Процедуры и макрокоманды развертываются в теле спецификации. Детали трансляции представлены в [6].

Каждый экземпляр процесса может порождать или уничтожать экземпляры процессов в своем блоке. Подобно языку SDL, в языке dREAL количество экземпляров, существующих с самого начала, и максимально возможное количество экземпляров определяются в заголовке процесса.

Одним из основных отличий языка SDL от языка dREAL является различный механизм передачи сигналов от процесса к процессу. В языке SDL все входные сигналы хранятся в порядке их поступления в едином буфере процесса, а не в том канале, в который они поступили, как в языке dREAL. При трансляции в каждом блоке dREAL-спецификации создается процесс-буфер. Все каналы от одного процесса к другому перенаправляются в процесс-буфер, а из него выходит один канал до целевого процесса. Процесс-буфер читает приходящие сигналы и записывает их в этот канал. Таким образом, в единственном входном канале процесса все входящие сигналы расположены в порядке их поступления в процесс-буфер. В случае одновременного поступления сигналов они располагаются во входном канале процесса в случайном порядке, как и в языке SDL.

В языке SDL сигнал удаляется из очереди, если не предусмотрена его обработка в текущем состоянии. В этом случае процесс пытается обработать следующий сигнал. В языке dREAL в этом случае процесс переходит в режим ожидания сигнала. При трансляции в каждом состоянии процесса добавляется обработка всех сигналов, которые могут приходить к нему. Если приходит сигнал, для которого в этом состоянии в исходной SDL-спецификации не предусмотрена обработка, то в результирующей dREAL-спецификации этот сигнал удаляется из канала.

Рассмотрим оператор SAVE языка SDL, который позволяет читать не первый пришедший сигнал, а следующий за ним, сохраняя при этом первый сигнал в общей очереди. Поскольку в языке dREAL такой конструкции нет, то моделируется поведение оператора SAVE. Все приходящие сигналы сохраняются в массиве, который моделирует общий буфер SDL-процесса. Далее процесс читает сигнал из этого массива по текущему индексу. В случае, если приходящий сигнал нужно сохранить в данном состоянии, процесс обрабатывает следующий сигнал по индексу, увеличенному на 1. При переходе в следующее SDL-состояние индекс, по которому процесс читает сигналы из массива, обнуляется, и процесс обрабатывает первый сигнал в очереди.

В языке SDL оператор PROVIDED <условие> позволяет принимать сигнал при выполнении этого условия. Если условие не выполняется, то сигнал сохраняется. Поэтому при трансляции в dREAL создается два альтернативных перехода с условным dREAL-оператором WHEN: первый переход для случая, когда условие выполнено, и тогда используется оператор READ для чтения из канала, и второй переход для случая, когда условие не выполнено, и тогда реализуется механизм для оператора SAVE.

3. Система моделирования dREAL-спецификаций

3.1. Язык запросов

Для языка запросов использованы конструкции языка SDL с добавлением служебных слов. Структура запроса представляет собой конструкцию:

Квантор Объект{ If (условие) операция; }[Период]

Квантор может быть представлен квантором всеобщности All или существования Ex. Объект – это экземпляр процесса. Условие может представлять собой как отдельный предикат, так и конъюнкцию предикатов. Предикат может быть представлен отношением между переменными и параметрами, локатором контроля управления, контроллером пустоты или полноты канала, чекером присутствия сигнала в канале. Операция представляется с помощью следующих операторов: STOP – остановить процесс симуляции, PAUSE – приостановить процесс симуляции, PRINTLOG («сообщение») – записать «сообщение» в протокол работы системы. Период – это отрезок времени, за который система должна выполнить запрос и остановиться или остановить работу без выполнения запроса по истечении заданного времени.

3.2. Система моделирования

Система состоит из синтаксического анализатора, модуля моделирования, конструктора графа переходов и модуля обратного хода.

Система моделирования позволяет провести проверку ряда свойств dREAL-спецификации. Эта спецификация и запросы, выражающие ее свойства, передаются синтаксическому анализатору, который переводит их во внутреннее представление и передает модулю моделирования. Затем на каждом шаге рассматриваются все возможные состояния системы. Для каждого состояния получается конфигурация перехода, которая содержит значения переменных, состояния каналов и состояния процессов для этого шага моделирования. Модуль моделирования передает каждую новую конфигурацию конструктору графа переходов. На каждом шаге модуль моделирования проверяет истинность свойств, заданных запросами. Выполнение каждой ветки происходит до тех пор, пока не выполнится запрос, либо система не придет в завершающее состояние, либо не закончится время, заданное в запросе. Таким образом, модуль моделирования совершает полный обход спецификации во всех возможных конфигурациях. По окончании работы система сохраняет граф переходов.

Поскольку исходная спецификация представлена на языке SDL, то и результаты моделирования желательно представить в языке SDL. Модуль обратного хода предназначен для отображения в исходной SDL-спецификации результатов, полученных в ходе моделирования. В том случае, если в запросе используются только предикаты отношения между переменными или локаторы контроля состояния процесса, обратный ход не представляет сложности, поскольку имена исходных переменных, процессов и состояний совпадают с соответствующими именами в dREAL спецификации. В случае, если в dREAL запросе используются контроллеры пустоты / полноты канала или чекеры присутствия сигнала в канале, то анализируется, к какому процессу направлен канал. В SDL сигналы хранятся в буфере процесса, который принимает сигналы. Значит, при обратном ходе из dREAL в SDL процесс, к которому направлен канал, и будет хранить сигналы.

3.3. Конвертер запросов (Конвертер2)

Конвертер запросов переводит запросы из терминов SDL в термины dREAL. При этом имена переменных, состояний и процессов, использующихся в запросах, сохраняются без изменений, контроллеры пустоты / полноты буфера переходят в контроллеры пустоты / полноты канала от процесса-буфера в процесс, чекер присутствия сигнала в буфере переводится в чекер присутствия сигнала в канале от процесса-буфера к процессу.

4. Верификация SDL-спецификаций

Для верификации SDL-спецификаций используются Транслятор1, переводящий SDL-спецификацию в dREAL-спецификацию, и Транслятор2, переводящий ее в Promela-спецификацию для использования в системе SPIN. Свойства SDL-спецификаций переводятся в LTL-формулы при помощи Конвертера1 (см. рис. 1).

Текущая версия транслятора из языка dREAL в язык Promela ориентирована на использование современной версии SPIN [8].

Временные конструкции. Поскольку в Promela отсутствуют встроенные средства для моделирования временных интервалов, для их моделирования в выходную Promela-спецификацию добавляется глобальная переменная *TICK* и специальный процесс clock, который увеличивает ее значение.

Статические конструкции. Как и в SDL и dREAL, в языке Promela есть переменные, массивы, каналы и процессы. Чтобы избежать коллизии имен, состояния (метки в Promela), каналы и сигналы языка dREAL при переводе получают префиксы.

При посылке сигнала в канал добавляется параметр**,** означающий в языке Promela идентификатор данного экземпляра процесса, а значение этого параметра при чтении сигнала сохраняется в переменной *aSENDER*.

Процедуры. Процедуры dREAL транслируются в inline-макроопределения языка Promela. В процедурах могут использоваться локальные переменные, для которых (локальных переменных) используются Promela-переменные с ключевым словом local.

Динамические конструкции. В языке Promela число одновременно существующих экземпляров процессов не может превосходить 255, поэтому такое же ограничение должно выполняться для языка dREAL.

В языке dREAL уничтожение экземпляра процесса осуществляется оператором STOP в уничтожаемом процессе. В системе SPIN можно уничтожать только последний созданный экземпляр процесса, причем его PID будет переиспользован при создании следующего экземпляра процесса. Поэтому процессы языка Promela, завершившие работу, не уничтожаются, а останавливаются в специальном состоянии. Это позволяет обеспечить уникальность идентификаторов процессов в системе SPIN. В текущей версии (SRDSVer3) добавлена возможность повторного использования PID, а для идентификации экземпляров добавлены дополнительные идентификаторы в каждый экземпляр. Это позволяет ослабить ограничение на общее количество процессов за счет усложнения трансляции и описания формул.

Логические спецификации. Рассматриваются логические спецификации, построенные из предикатов с помощью логических операций и кванторов по экземплярам процессов (в начале формул), моментам времени, а также кванторов общности по поведениям выполнимой спецификации.

Оптимизация моделей. С целью сокращения размера модели можно использовать при трансляции ограничения на максимальное значение показаний часов, на общее количестве процессов и на размер тех каналов, которые были неограниченными в dREAL-спецификации.

5. Верификация системы управления сетью банкоматов

5.1. Обзор системы управления сетью банкоматов

Эта система состоит из банкоматов и сервера, к которому банкоматы обращаются за информацией о корректности карты, наличии средств на ней и списании суммы со счета. Каждый банкомат инициализируется с присвоением порядкового номера. Как только банкомат освободится или инициализируется, клиент из очереди подходит к банкомату и нажимает на кнопку начала сеанса. Банкомат сообщает о готовности и ожидает ввода карточки. Получив карту, банкомат запрашивает у клиента PIN-код, который отсылает на сервер для проверки корректности. Если введенный PIN-код неверен, карточка возвращается клиенту и сеанс работы заканчивается, счетчик неудачных попыток увеличивается. После трех неудачных попыток карта блокируется. Если PIN-код верен, то банкомат, в зависимости от вида операции, либо запрашивает сумму для снятия, либо запрашивает баланс, либо вводит сумму для пополнения счета. В первом случае, получив сумму, банкомат проверяет сумму на лимиты снятия через банкомат, затем запрашивает у сервера наличие требуемой суммы на счете. Если лимиты превышены или суммы на счете недостаточно, банкомат передает это сообщение клиенту и сеанс работы заканчивается. Если же денег достаточно, то сервер списывает сумму со счета. Клиент получает деньги и чек. Если клиент выбрал пополнение счета, то банкомат принимает деньги,

передает информацию на сервер и выдает клиенту чек. В случае простого просмотра баланса терминал выдает информацию о счете на экран. В конце сеанса клиент завершает свою работу, а банкомат переходит в режим ожидания нового клиента из очереди.

5.2. Эксперименты с системой моделирования

Эксперименты проводились с использованием следующих данных: к трем банкоматам подходят пять клиентов, каждый из которых хочет снять определенную сумму денег со своей банковской карточки. Был проведен эксперимент со спецификацией, в которую намеренно внесли ошибку. В месте проверки введенной клиентом суммы строгое неравенство было заменено на нестрогое. В результате, если клиент запрашивает всю сумму со счета, то не получает ее.

В экспериментах полученная dREAL-спецификация и соответствующий запрос пользователя были поданы на вход системы моделирования. Целью эксперимента были моделирование выполнения спецификации и проверка ее основных свойств с помощью запросов пользователя.

Понятия, использующиеся в запросах: *Client[PID]* – экземпляр процесса Client, *wants_summ[]* – запрашиваемая клиентом сумма денег; *gotten_summ[]* – полученная сумма денег. Состояния экземпляра процесса Client: *Client[PID].good*, означающее, что требуемая сумма получена; *Client [PID].term_no_money* – в терминале недостаточно средств.

В процессе эксперимента были проверены следующие запросы.

1. В любой конфигурации запрашиваемая сумма денег больше единовременного лимита, но клиент не получит эту информацию и может даже получить деньги.

All PROCESS *Client,PID* { if((*Client[PID]* AT *Client[PID].look* && *wants_summ[PID]* > *ONE_TIME_LIMIT* && *correct_code* IN BUF *Client[PID]*) => (*Client[PID]* AT *Client[PID].good*)) PAUSE; }

2. Существует конфигурация, в которой запрашиваемая сумма денег корректна, карта корректна и код верен, но клиент не получит деньги.

Ex PROCESS *Client, PID* { if((*Client[PID]* AT *Client[PID].look* && *wants_summ[PID]* <= *ONE_TIME_LIMIT* && *correct_code* IN BUF *Client[PID]*) => NOT(*Client[PID]* AT *Client[PID].good* || *Client[PID]* AT *Client[PID]. term_no_money*) PAUSE; }

Если один из этих запросов выполняется, то SDL-спецификация содержит ошибки.

Входными данными для моделирования работы системы является информация о клиенте: номер его карточки, пин-код, запрашиваемая сумма и вид операции. Эксперимент проводился с пограничными наборами данных и со случайными.

Также был проведен *эксперимент с некорректной спецификацией*. При возникновении ошибочной ситуации система позволила проследить путь исполнения спецификации в исходной SDL-спецификации.

5.3. Эксперименты с системой SPIN

Эксперименты также проводились с шестью банкоматами и различным числом клиентов с исходными данными, описанными в начале этого раздела.

Также были проведены эксперименты с большим количеством клиентов, превышающим обычные ограничения для количества процессов. В обычном случае все созданные процессы после своего завершения принудительно останавливаются в специальном завершающем состоянии «__end_of_process», чтобы можно было однозначно определить экземпляр процесса по его PID. Но это ограничивает общее количество созданных процессов (даже если число одновременно выполняющихся процессов не очень велико). Для преодоления этого недостатка была введена дополнительная переменная *LPID* (от Long Process Identifier), которая однозначно идентифицирует экземпляр процесса.

Из-за того, что в системе SPIN в формулах не используются локальные переменные, эти переменные отмечены в SDL-спецификации как REVEALED, что делает их глобальными массивами при

переводе в dREAL. Таким образом, локальный массив SDL-процесса, не имеющего экземпляров, становится глобальным массивом, а скалярная переменная SDL-процесса с экземплярами становится в dREAL глобальным массивом, индексированным идентификаторами процессов.

В силу специфики системы SPIN для верификации используются отрицания проверяемых формул, для которых SPIN находит контрпримеры.

Так как язык SPLan содержит кванторы по экземплярам процессов, а язык временной логики LTL не имеет кванторов по переменным, то заданное свойство, содержащее такие кванторы, проверяется для каждого экземпляра процесса.

Для представления свойств были определены следующие функции и предикаты.

Функции: количество денег на счете клиента; количество денег в банкомате; суточный лимит выдачи в банкомате; количество денег, нужное клиенту; количество денег, полученное клиентом.

Предикаты: клиент получил сообщение о состоянии счета; на счете достаточно денег; в банкомате достаточно денег; суточный лимит выдачи не превышен; клиент получил некоторую сумму денег; клиент получил нужную ему сумму денег; пин-код правильный.

Были рассмотрены следующие свойства:

1) если клиенту нужна некоторая сумма, он знает пин-код и везде (на счете и в банкомате, и лимит не превышен) достаточно денег, то он получит нужную ему сумму;

2) если на счете клиента недостаточно денег, он не получит ничего;

3) если в банкомате недостаточно денег, клиент не получит ничего;

4) для «ошибочной спецификации», когда клиент пытается снять все свои деньги, ему это не удается (из-за нестрогого неравенства в ошибочной спецификации для сервера).

Подробные результаты можно найти в Bitbucket-репозитории [9].

С помощью системы SPIN для систем с шестью банкоматами и пятью клиентами была доказана истинность всех этих свойств, а для систем с шестью банкоматами и 100 клиентами – свойств 1, 2 и 3.

Заключение

Язык Dynamic-REAL является комбинированным языком спецификаций как распределенных систем, так и их свойств. Преимущества языка dREAL обусловлены синтаксисом, допускающим графическое представление выполнимых спецификаций, формальной операционной семантикой и выразительным языком представления свойств. Разработанный программный комплекс SRDSVer3 является мощной системой, которая включает трансляторы из SDL в dREAL и из dREAL во входной язык Promela системы верификации SPIN, системы симуляции и автоматического моделирования dREAL-спецификаций. Входной язык этого комплекса включает все базовые конструкции языка SDL, которые широко применяются на практике [10, 11]. Система автоматического моделирования играет важную роль, так как может успешно применяться и в тех случаях, когда система верификации SPIN неприменима из-за громоздких моделей SDL-спецификаций. Преимущества нашего подхода иллюстрируются автоматическим моделированием и верификацией динамической системы управления сетью банкоматов.

Программный комплекс SRDSVer3 является существенным развитием нашего комплекса, описанного в работе [6]. Комплекс SRDSVer3 позволяет пользователю задавать спецификации распределенных систем и их свойства, а также получать контрпримеры в терминах SDL.

Планируется дальнейшее развитие комплекса SRDSVer3 с целью верификации более широкого подмножества языка SDL-2010. Это предполагает как расширение языка dREAL, так и расширение систем данного комплекса. Также предполагается расширить язык запросов системы автоматического моделирования с целью обнаружения тупиков и зацикливания.

ЛИТЕРАТУРА

1. Specification and Description Language (SDL) : ITU-T Recommendation : Z.100. 2015.
2. Reed R. Data encoding for SDL in ITU-T Rec. Z.104 // Lecture Notes in Computer Science: System Analysis and Modeling. 2005. V. 3319. P. 80–95.

3. Reed R. SDL-2010: Background, rationale, and survey // Lecture Notes in Computer Science: SDL 2011: Integrating System and Software Modeling. 2012. V. 7083. P. 4–25.
4. Vlaovic B., Vreže A., Brezočnik Z. Applying automated model extraction for simulation and verification of real-life SDL Specification with Spin // IEEE Access. 2017. P. 1–12. DOI: 10.1109/ACCESS.2017.2685238
5. Bozga M., Graf S., Ober Il., Ober Iu., Sifakis J. The IF toolset // Lecture Notes in Computer Science: SFM-RT 2004: Formal Methods for the Design of Real-Time Systems. 2004. V. 3185. P. 237–267.
6. Непомнящий В.А., Бодин Е.В., Веретнов С.О. Язык Dynamic-REAL и его применение для верификации SDL-спецификаций распределенных систем // Программирование. 2014. Т. 40, No. 5. С. 34–44
7. Shi H., Ma W., Yang M., Zhang X. A Case study of model checking retail banking system with SPIN // Journal of Computers. 2012. V. 7. P. 2503–2510.
8. Holzmann G.J. The SPIN model checker. Primer and reference manual. Addison-Wesley, 2004. 596 p.
9. SRDSV3 examples // Bitbucket repository. URL: https://bitbucket.org/drealsdlan/SRDSV3
10. Grammes R., Gotzhein R. SDL Profiles – Formal Semantics and Tool Support // Lecture Notes in Computer Science: Fundamental Approaches to Software Engineering. 2007. V. 4422. P. 200–214.
11. Doldi L. Validation of communications systems with SDL. Wiley, 2003. 312 p.

Поступила в редакцию 20 марта 2020 г.

Nepomniaschy V.A., Bodin E.V., Veretnov S.O. (2020) THE ANALYSIS AND VERIFICATION OF SDL-SPECIFICATIONS OF DISTRIBUTED SYSTEMS USING DYNAMIC-REAL LANGUAGE. *Vestnik Tomskogo gosudarstvennogo universiteta. Upravlenie, vychislitelnaja tehnika i informatika* [Tomsk State University Journal of Control and Computer Science]. 53. pp. 118–126

DOI: 10.17223/19988605/53/12

At present, the role of formal methods used to develop distributed systems, such as control systems and communication protocols, is quite significant. SDL language is widely used to specify distributed systems. Therefore, the problem of analysis and verification of SDL-specifications of distributed systems is actual.

This problem has been considered in a number of publications where the use of the well-known SPIN system for verification of SDL specifications using IF language as the intermediate language is described. In contrast to that approach, along with the SPIN verification system, we use a system for modeling the behavior of SDL specifications, as well as a system for finding errors in SDL specifications with the help of the obtained counterexamples. This allows us to analyze the SDL-specifications even when the SPIN system is not applicable.

This paper describes the new software package SRDSVer3 (SDL / REAL Distributed Systems Verifier), intended for modeling, analysis and verification of SDL-specifications. The input language of the software package SRDSVer3 includes the basic base constructs of the SDL-2010 language, such as BLOCK, PROCESS, SIGNALROUTE, CHANNEL, SAVE, SIGNAL.

To describe the properties of SDL-specifications, an SDL Properties Language (SPLan) was developed, based on linear time logic (LTL), extended by quantifiers for process instances. For modeling and analysis of SDL-specifications, a query language was developed.

This software package uses the intermediate specification language Dynamic-REAL (dREAL), which is a combined specification language for both distributed systems and their properties. The advantages of dREAL are caused by its syntax for textual and graphical presentation of specifications, formal operational semantics and expressive language for representing properties.

The developed software package SRDSVer3 is a powerful tool that includes a translator from language SDL to language dREAL and two tools for analysis and verification. The first tool includes a translator from language dREAL to the input language Promela of the SPIN verification system, a converter of properties of SDL-specifications to LTL formulas, and a SPIN verification system. The second tool consists of a tool for simulation and automatic modeling of dREAL specifications, and a converter for translating queries expressed in SDL into dREAL. The automatic modeling tool plays an important role, since it can be successfully applied in the cases when the SPIN verification system is not applicable due to the large size of the SDL-specification models. The input language of software package SRDSVer3 includes the basic base language constructions of language SDL-2010 widely used in practice. The package SRDSVer3 allows the user to verify specifications of distributed systems with respect to their properties represented in the language SPLan, as well as to obtain the counterexamples in terms of SDL.

The advantages of this approach are illustrated by automatic modeling and verification of a dynamic ATM network management system. An experiment was also conducted with an incorrect specification of this management system, where an error was introduced at the level of the SDL specification. If an error is detected, the package allows to trace the execution path in the original SDL specification.

Keywords: distributed systems; SDL language; Dynamic-REAL language; modeling analysis; verification; Promela language; SPIN verifier.

NEPOMNIASCHY Valery Aleksandrovich (Candidate of Physical and Mathematical Sciences, Siberian Division of the Russian Academy of Sciences A.P. Ershov Institute of Informatics Systems, Novosibirsk, Russian Federation).
E-mail: vnep@iis.nsk.su

BODIN Evgeniy Viktorovich (Researcher, Siberian Division of the Russian Academy of Sciences A.P. Ershov Institute of Informatics Systems, Novosibirsk, Russian Federation).
E-mail: bodin@iis.nsk.su

VERETNOV Sergey Olegovich (Junior Researcher, Siberian Division of the Russian Academy of Sciences A.P. Ershov Institute of Informatics Systems, Novosibirsk, Russian Federation).
E-mail: veretnovs@mail.ru

REFERENCES

1. ITU-T (2015). Specification and description language (SDL). *ITU-T Recommendation*, Z.100.
2. Reed, R. (2005) Data encoding for SDL in ITU-T Rec. Z.104. *Lecture Notes in Computer Science: System Analysis and Modeling.* 3319. pp. 80–95. DOI: 10.1007/978-3-540-31810-1_6
3. Reed, R. (2012) SDL-2010: Background, rationale, and survey. *Lecture Notes in Computer Science: SDL 2011: Integrating System and Software Modeling.* 083. pp. 4–25. DOI: 10.1007/978-3-642-25264-8_3
4. Vlaovic, B., Vreže, A. & Brezočnik, Z. (2017) Applying automated model extraction for simulation and verification of Real-Life SDL specification with spin. *IEEE Access.* pp. 1–12. DOI:10.1109/ACCESS.2017.2685238
5. Bozga, M., Graf, S., Ober, Il., Ober, Iu. & Sifakis, J. (2004) The IF toolset. *Lecture Notes in Computer Science: SFM-RT 2004: Formal Methods for the Design of Real-Time Systems.* 3185. pp. 237–267. DOI: 10.1007/978-3-540-30080-9_8
6. Nepomnyashchy, V.A., Bodin, E.V. & Veretnov, S.O. (2015) The language dynamic-real and its application for verification of SDL-specified distributed system. *Programmirovanie – Programming and Computer Software*. 40(5). pp. 34–44. DOI: 10.1134/S0361768815010053
7. Shi, H., Ma, W., Yang M., & Zhang, X. (2012) A case study of model checking retail banking system with SPIN. *Journal of Computers.* 7. pp. 2503–2510. DOI:10.4304/jcp.7.10.2503-2510
8. Holzmann, G.J. (2004) *The SPIN model checker. Primer and reference manual*. Addison-Wesley.
9. SRDSVer3 examples – Bitbucket repository: https://bitbucket.org/drealsdlan/SRDSVer3
10. Grammes, R. & Gotzhein, R. (2007) SDL Profiles-formal semantics and tool support. *Lecture Notes in Computer Science: FundamentalApproaches to Software Engineering.* 4422. pp. 200–214. DOI: 10.1007/978-3-540-71289-3_17
11. Doldi, L. (2003) *Validation of communications systems with SDL*. Wiley.