УДК 004.92+004.94

И.С. Анурeев, С.Н. Баранов, Д.М. Белоглазов,
Е.В. Бодин, П.Д. Дробинцев, А.В. Колчин,
В.П. Котляров, А.А. Летичевский, А.А. Летичевский мл.,
В.А. Непомнящий, И.В. Никифоров, С.В. Потиенко,
Л.В. Прийма, Б.В. Тютин

СРЕДСТВА ПОДДЕРЖКИ ИНТЕГРИРОВАННОЙ ТЕХНОЛОГИИ ДЛЯ АНАЛИЗА И ВЕРИФИКАЦИИ СПЕЦИФИКАЦИЙ ТЕЛЕКОММУНИКАЦИОННЫХ ПРИЛОЖЕНИЙ

Анурeев И.С., Баранов С.Н., Белоглазов Д.М., Бодин Е.В., Дробинцев П.Д., Колчин А.В., Котляров В.П., Летичевский А.А., Летичевский А.А. мл., Непомнящий В.А., Никифоров И.В., Потиенко С.В., Прийма Л.В., Тютин Б.В. **Средства поддержки интегрированной технологии для анализа и верификации спецификаций телекоммуникационных приложений**

Аннотация. В работе описываются разработанные авторами инструментальные средства и комплексный подход на их основе, при котором методы и средства анализа и верификации обеспечены для представителей всех четырех основных классов языков, на которых обычно описываются телекоммуникационные приложения: языки выполняемых спецификаций общего назначения (SDL), языки для описания и анализа укрупненных образцов поведения и выявления зависимостей между ними в сложных системах (UCM), специализированные языки, ориентированные на верификацию спецификаций телекоммуникационных систем (язык интерпретированных MSC диаграмм, язык взаимодействующих конечных автоматов, язык Dynamic-REAL) и индустриальные императивные языки (C/C++).

Верификация спецификаций дополняется автоматизированным построением тестовых наборов, обеспечивающих заданную степень покрытия исходных поведенческих требований, причем эти тестовые наборы оптимизированы по заданным критериям производительности. Исполнение тестов происходит в среде автоматизированного тестирования на моделях систем, либо непосредственно на их реализациях, погруженных в соответствующие программные оболочки, обеспечивающие взаимодействие тестируемой системы с тестовым окружением. Тестовая оболочка позволяет одновременно с прогоном тестов проводить автоматизированный анализ результатов тестирования.

Ключевые слова: базовые протоколы, верификация, ключевые агенты, операционная семантика, логика безопасности, телекоммуникационные системы, помеченные системы переходов, язык спецификаций SDL, язык спецификаций Dynamic-REAL, система верификации SPIN, системы конечных автоматов, раскрашенные сети Петри.

Anureev I.S., Baranov S.N., Beloglazov D.M., Bodin E.V., Drobitsev P.D., Kolchin A.V., Kotlyarov V.P., Letichevsky A.A., Letichevsky A.A. jr., Nepomniaschy V.A., Nikiforov I.V., Potiyenko S.V., Priyma L.V., Tyutin B.V.

Tools of Integrated Technology for Analysis and Verification of Telecom Application Specs

Abstract. The paper describes an integrated approach and respective tools which provide methods and means for analysis and verification for representatives of all four major classes of languages used to describe telecom applications: languages of general purpose executable specifications (SDL), languages for high-level descriptions of complex systems behavior patterns and their interconnections (UCM), special purpose language for verification of telecom applications (interpreting MSC, communicating finite state machines, Dynamic-REAL), and industrial imperative languages (C/C++).

Verification of specifications is complemented with automated construction of test suites, which ensure the specified test coverage rate of source behavioral specifications and are optimal with respect to specified performance criteria. Tests are run in a specialized automated test-run environment either on system models, or on actual system implementations inserted in respective program shells which provide communication of the system under test with the test environment. The test shell allows for automated analysis of the test-run results along with the test-runs as well.

Keywords: basic protocols, verification, key agents, operational semantics, safety logic, telecommunication systems, labelled transition systems, specification language SDL, specification language Dynamic-REAL, verification system SPIN, finite automata systems, coloured Petri nets.

---

1. Введение. В настоящее время уже создано и и активно развивается довольно много специализированных систем верификации, среди которых можно выделить наиболее популярные – SPIN [33], NuSMV [37], VeriSoft [39], BLAST [34], CBMC [36], VCEGAR [48]. Все они так или иначе решают задачу верификации спецификаций программных систем, различаясь, прежде всего, своим входным языком представления формальных моделей, ограничениями и масштабируемостью. В то же время ни одна из них пока не стала системой, применяемой в промышленности.

Данная работа описывает ряд инструментальных средств и систем, разработанных до уровня демонстрационных прототипов в рамках единой интегрированной технологии анализа и верификации спецификаций, отдельные компоненты которой создавались на протяжении ряда лет тремя группами исследователей. Группа, работающая в Киеве под руководством академика А.А. Летичевского, создала мощный верификатор на базе теории инсерционного программирования и языка АПЛАН. Этот верификатор гибко соединяет метод поиска логического вывода в алгебраической теории с равенствами и неравенствами с методом проверки на моделях, который строит и анализируерт поведенческие трассы системы,

описываемой набором ее базовых протоколов – элементарных сценариев в описании поведения системы на заданном уровне абстракции. Основная проблема метода проверки на моделях – «комбинаторный взрыв» достижимых состояний верифицируемой модели – решается управляемым комбинированием статических и динамических методов проверки, реализованных в технике символьного и конкретного моделирования.

Коллектив, работающий в Санкт-Петербурге под руководством проф. В.П. Котлярова, разработал инструментальные средства для автоматизации тестирования программных приложений, в котором из формальной модели приложения порождаются тестовые наборы и среда тестирования, обспечивающие покрытие исходных требований в соответствии с заданными критериями покрытия, причем сам тестовый набор может оптимизироваться по числу входящих в него тестов или по их суммарной длине (под тестом здесь понимается его поведенческая трасса). Входным языком для описания исходных поведенческих требований является графический язык пользовательский отображений (UCM — Use Case Maps), сочетающий наглядность представления с высоким уровнем абстракции при задании поведенческих схем. Инструментальное средство UCM2BP преобразует описания поведения на языке UCM в соответствующие им базовые протоколы, после чего работает верификатор базовых протоколов, порождающий поведенческие трассы, из которых по специальной методике отбираются те из них, которые затем преобразуются в тестовые примеры.

Наконец, группа, работающая в Новосибирске под руководством проф. В.А. Непомнящего, создала верификатор для спецификаций на языке SDL (Specification and Description Language) для распределенных систем на базе языка dREAL, использующий известный верификтор общего назначения SPIN, верификатор автоматных спецификаций для систем взаимодействующих расширенных конечных автоматов и систему верификации Спектр-К на базе языка спецификаций Atoment для верификации телекоммуникационных систем, представленных на языке C.

Объединение результатов всех этих трех групп в рамках единой интегрированной технологии дает перспективный и мощный инструмент для анализа и верификации приложений с существенно динамическим поведением, к которым относятся прежде всего телекоммуникационные приложения. Дальнейшее развитие описы-

ваемых результатов состоит в превращении этих прототипов в программные продукты промышленного качества.

2. Формальные модели и их свойства. Базой для применения описываемой технологии является формальная модель приложения, которая строится вручную по исходным неформальным требованиям, описывающим его поведение, с применением средств автоматизации. После того, как модель построена, к ней применяются описанные ниже инструментальные средства, устанавливающие выполнимость определенных ее свойств (непротиворечивость, полнота и другие) и в случае корректной модели создающие тестовые наборы, гарантированно покрывающие исходные требования в соответствии с заданным критерием покрытия.

Язык формального описания моделей. Спецификация формальной модели состоит из описания среды, включающего декларацию агентов и типизированных атрибутов, переходов и проверяемых свойств. Агенты работают параллельно, асинхронно и взаимодействуют между собой посредством чтения и изменения атрибутов. Состояние среды состоит из набора значений ее атрибутов и значений атрибутов всех действующих агентов. Переходы между состояниями модели записываются на языке «базовых протоколов» [12]. Базовый протокол представляет собой выражение вида тройки Хоара: $\alpha \rightarrow< u > \beta$, где α — предусловие, u — процесс (действие), β — постусловие, и описывает атомарный параметризированный (имя агента — один из параметров) переход модели. Пред- и постусловия являются формулами логики первого порядка. Постусловие так же может содержать операторы присваивания.

Процесс базового протокола представляет собой линейную последовательность MSC конструкций, в терминах которых часто специфицируют телекоммуникационные приложения. Преимущество этого формализма так же в простоте и эффективности использования методов верификации – язык использует базовый набор структур данных и операций над ними, и при этом обладает достаточной выразительной мощностью для описания систем, специфицированных в таких инженерных языках, как MSC, SDL, UML [23, 24].

Проверяемые свойства формальных моделей. В описываемой системе верификации осуществляется проверка таких свойств: недетерминизм, выход за пределы допустимых значений, переполнение (over/underflow), обращение к неинициализированному эле-

менту системы, а так же проверка условий безопасности, достижимости и отсутствия тупиков (deadlock, livelock). Всякий раз, когда достигнуто состояние, в котором нарушается одно из перечисленных свойств, будет построена трасса, которая иллюстрирует поведение модели из ее начального состояния в текущее.

Трасса представляет собой последовательность имен переходов модели (т.е. имен базовых протоколов с их фактическими параметрами), а так же может быть построена из событий (MSC конструкций), заданных в процессной части базовых протоколов. Такие события описывают наблюдаемое поведение модели и могут быть использованы в качестве сценарных тестовых контр-примеров.

Недетерминизм в модели является не ошибкой ее поведения, а лишь сигналом для дальнейшего анализа. Важно отделить естественный недетерминизм от ошибочного. Некоторые системы верификации задают недетерминизм путем введения специальных атрибутов, значения которых выбираются недетерминировано в процессе моделирования.

Выход за пределы допустимых значений – распространенная ошибка, часто являющаяся источником нарушения безопасности систем. Проверка обеспечивается путем генерации дополнительных ограничений на индексацию в пред- и постусловиях переходов (обращение к элементу массива, параметризированному атрибуту, либо при обращении к процессу по индексу).

Переполнение (over/underflow) определяется таким образом: арифметические выражения, согласно приоритетам операций, раскладываются на эквивалентную последовательность вычислений значений бинарных операций, вводятся вспомогательные локальные переменные для хранения и дальнейшего использования результата предыдущих вычислений, и перед каждым производится проверка на возможное переполнение.

Проверка использования неинициализированного атрибута модели обеспечивается посредством (неявного) введения вспомогательных атрибутов, цель которых — хранить информацию об инициализации каждого атрибута модели и генерировать дополнительные проверки таких атрибутов. Если производится присваивание, в правой части которого есть неинициализированный атрибут, то переход осуществляется, а атрибут, стоящий в левой части присваивания становится неинициализированным.

Условия безопасности (safety, liveness) выражаются формула-

ми линейной темпоральной логики (LTL – Linear Temporal Logic). В качестве примеров можно привести такие свойства: «лифт никогда не едет с открытой дверью», «выделенный ресурс когда-нибудь освободится», и т.д.

Проверка достижимости ввыполняется как для целевых состояний, так и для переходов. Целевое состояние формулируется пользователем в виде LTL-формулы. При окончании работы генерируется вердикт, включающий в себя информацию о статистике выполнения переходов.

Тупиковой (deadlock) называется ситуация, из которой нет ни одного допустимого перехода.

Замкнутой (livelock) называется ситуация, из которой нет возможности достигнуть хотя бы одно состояние, достижимое из начального (initial) состояния. Другими словами, условие отсутствия livelock можно сформулировать так: каждое достижимое состояние должно быть достижимо из любого изначально достижимого состояния. Проверяется так же, как и «сильная связность» в терминах теории графов.

3. Инструментарий проверки свойств. Разработаны методы символьного и конкретного моделирования (так называемые «трассовые генераторы»), а так же методы статической проверки.

Конкретный трассовый генератор используется, когда состояния модели конкретные. Ядром трассовых генераторов является классический поиск в глубину (опционально в ширину), с некоторыми специфическими конфигурациями (выполняющими роль ограничителей) и оптимизациями. Далее представлены методы оптимизации обхода пространства поведения верифицируемой модели.

- **Создание хеш-таблиц переходов модели.** Наиболее удачной хеш-функцией на практике является разбиение по принципу проверки потока управления. Для каждого процесса такой атрибут должен явно сравниваться в предусловиях всех переходов и только с конкретными значениями; он отделяется от общей формулы предусловия и называется «ключевым состоянием». Таким образом, список «кандидатов» на проверку допустимости существенно сокращается. На практике трансляции UML-автоматов в язык базовых протоколов описанная оптимизация на каждом состоянии сокращает чис-

ло потенциальных переходов для каждого процесса в среднем с нескольких тысяч до одного-трех. См. также [9,21].

- **Элиминация избыточного интерливинга.** Проектирование и тестирование параллельных систем усложняется тем, что компоненты могут взаимодействовать множеством способов. Интерливинг (перестановка выполнения действий параллельно работающих компонентов) является одним из источников комбинаторного взрыва числа вариантов поведения верифицируемой модели. Актуальной задачей является ограничение их числа путем исключения незначимых с точки зрения проверяемых свойств перестановок. Параллельные системы состоят из множества асинхронно работающих сообщающихся между собой и внешней средой процессов. Точки сообщения (доступ к общим ресурсам) могут быть определены статически, основываясь на структурном описании верифицируемой модели. Аналогично [38,47], исследуя переходы, рассматриваются лишь частичные перестановки переходов. Заметим, что такой подход неприменим для символьного моделирования, так как можно задать состояние (начальное, либо построить динамически в процессе обхода), включающее условие зависимости атрибутов разных процессов, при этом оставляя их синтаксически «локальными».

- **Перестановка экземпляров процессов при проверке эквивалентности состояний.** Часто на практике встречаются модели, содержащие множество однотипных процессов, например, телекоммуникационные задачи, в которых участвуют N телефонов. Аналогично методам симметрии [40], функция проверки эквивалентности состояний модифицируется добавлением проверок дополнительных состояний, порожденных путем перестановок атрибутов модели по принципу их принадлежности к процессам, а так же хранению значения идентификаторов процессов, таким образом, ослабляя критерий эквивалентности.

- **Трансляция формализации в языки низкого уровня.** Описание среды, пред- и постусловий переходов, а так же функции проверки необходимости интерливинга, функции, порождающей перестановки атрибутов для ослабленной эк-

вивалентности, транслируются в C-функции и компилируются вместе с ядром верификатора в бинарный исполняемый код. Предварительная трансляция формального описания модели в язык C повышает производительность обхода пространства поведения в тысячи раз по сравнению с методами, основанными на интерпретации.

- **Редукция пространства генерируемых состояний модели.** Это достигается за счет применения метода динамической абстракции состояний модели, что позволяет игнорировать некоторые незначимые с точки зрения проверяемых свойств детали; таким образом, два различных состояния могут рассматриваться как одинаковые. Существуют примеры моделей, на которых применение метода дает значительное (комбинаторное) сокращение затрат времени и памяти, необходимых для проверки свойств [7].

Производительность с учетом описанных оптимизаций возросла в десятки тысяч раз, и составила 300 тысяч состояний в секунду на платформе Pentium IV 3.0 GHz с 1GB оперативной памяти. Авторы работают над дальнейшем совершенствованием методов абстракции предикатов и других методов редукции пространства поиска.

Символьный трассовый генератор работает с состояниями модели, представленными формулами. Для систем с типами данных, включающими большой диапазон значений, одна символьная трасса может покрывать множество (возможно, бесконечное) конкретных трасс. Это свойство особенно эффективно используется при верификации моделей, использующих числовые типы данных. На каждом шаге символьного моделирования вместо вычисления набора конкретных значений атрибутов строится логическая формула, покрывающая некоторое множество конкретных состояний. Это множество далее будем называть символьным состоянием. Например, $\exists x(f(x) = 0) \wedge (f(a) > b)$.

Базовый протокол с предусловием α применим в состоянии s, если выполнима следующая формула: $s \wedge \alpha$.

Применимый базовый протокол осуществляет переход модели, а новое состояние s' задается формулой, построенной с помощью предикатного трансформера (pt) [11]: $s'{=}pt(s \wedge \alpha, \beta)$, где β — постусловие протокола, с помощью которого изменяются значения атрибутов.

Так как проверка применимости связана с вызовом разрешающих процедур, зачастую имеющих экспоненциальную сложность, используются методы быстрого отсечения неприменимых протоколов, такие как хеширование предусловий по конкретным значениям проверяемых атрибутов [21], а так же статическое определение отношения следования над базовыми протоколами.

Так как символьные трассы построены с применением экзистенциального моделирования, в качестве параметров событий могут выступать не конкретные значения, а формулы, то в своем исходном виде такие трассы не пригодны для создания по ним исполнимых тестов. Для устранения этого недостатка разработан [6] модуль конкретизации символьных трасс и процесс конкретизации полностью автоматизирован, причем выбор конкретных значений интеллектуален: технология позволяет контролировать покрытие граничных значений параметров теста, что в результате дает возможность повысить качество создаваемого тестового набора.

Статическая проверка свойств. В некоторых случаях верификацию модели можно осуществить, исследуя взаимоотношения между условиями верификации и базовыми протоколами, избегая порождения трасс. В частности, применяя базовые протоколы к символьному состоянию, определенному проверяемым условием безопасности, можно проверять инвариантность такого условия. Статически можно также доказывать детерминированность поведения системы и отсутствие тупиков [22].

Поскольку каждый базовый протокол параметризирован агентом, то практическую ценность имеет поиск недетерминированного поведения для каждого агента отдельно. Состояние s какого-либо агента a называется недетерминированным, если этот агент может осуществить более одного перехода из этого состояния, т.е. в состоянии s применим более чем один протокол с ключевым агентом a.

Введем понятие непротиворечивости для систем базовых протоколов. Оно определяется отсутствием пересечений предусловий базовых протоколов для каждого ключевого агента. Пусть $\alpha(a, x, r)$ и $\alpha'(a, y, r)$ — предусловия двух различных базовых протоколов b и b' с ключевыми агентами одного и того же типа, конкретизированные одним и тем же именем ключевого агента (в случае, если это имя является параметром). Списки x и y — это списки параметров, r — список атрибутных выражений. Протоколы b и

b' непротиворечивы, если формула $\neg(\exists x(\alpha(a,x,r)) \wedge \exists y(\alpha'(a,y,r)))$ тождественно истинна (отрицание невыполнимо). Система базовых протоколов называется непротиворечивой, если любая пара протоколов с однотипными ключевыми агентами непротиворечива. Для распознавания непротиворечивости системы базовых протоколов в среде, имеющей n типов агентов, поведение каждого из которых описано k_i $(i = 1, \ldots, n)$ базовыми протоколами, требуется $\sum_{i=1}^{n} C_2^{k_i}$ проверок непротиворечивости пар протоколов [11].

Если система базовых протоколов непротиворечива, то в каждом состоянии существует не более одного применимого базового протокола, а следовательно, все агенты такой системы имеют детерминированное поведение. Помимо детерминизма, критическим требованием к моделируемой системе является отсутствие тупиков. Введем понятие полноты для систем базовых протоколов. Оно определяется следующим условием: дизъюнкция условий применимости всех базовых протоколов $\exists x_1 \alpha_1(x_1, r_1) \vee \exists x_2 \alpha_2(x_2, r_2) \vee \ldots$ должна быть тождественно истинной (отрицание невыполнимо). Здесь $\alpha_i(x_i, r_i)$ — предусловие базового протокола, параметризированного списком переменных x_i и использующего список r_i атрибутных выражений.

Если выполняется условие полноты и в любой момент времени для каждого типа агентов в системе найдется агент этого типа, то в каждом состоянии системы найдется хотя бы один применимый протокол. Таким образом, в полной системе тупиков нет [11].

Для многоагентных систем, в которых задано разбиение базовых протоколов по типам ключевых агентов, можно сначала проверять полноту для каждого агента (типа агентов) отдельно, что существенно сокращает размер формул в доказательствах. Если неполнота найдена для всех типов агентов, то можно перейти к исследованию полноты системы в целом с помощью формул, представленных выше. Если в системе существует хотя бы один тип агентов, для которого доказана полнота, и в системе всегда есть по крайней мере один агент данного типа, то такая система не имеет тупиков, так как агенты этого типа всегда смогут осуществлять переходы.

Рассмотрим свойства, которые можно записать в виде формул базового языка и поставим требование истинности этих формул на всех состояниях системы. Назовем их условиями целостности. Построим алгоритм проверки условия целостности для системы

базовых протоколов. Пусть $Q(r)$ — формула базового языка, задающая условие целостности, $s_0(r)$ — формула базового языка, характеризующая начальное состояние системы (r — список атрибутных выражений). В первую очередь необходимо проверить целостность начального состояния системы: $s_0 \to Q(r)$. Эта формула должна быть общезначимой, иначе начальное состояние нарушает условие целостности, тогда нет смысла продолжать проверку и алгоритм должен завершить работу. Далее следует проверить инвариантность условия целостности. Для этого достаточно проверить для каждого протокола следующее: если протокол применим в состоянии, на котором условие целостности истинно, то после применения протокола это условие также будет истинным. Формально требуется проверить общезначимость следующей формулы: $\forall x(pt(\alpha(x,r) \wedge Q(r), \beta(x,r)) \to Q(r))$, где $\alpha(x,r)$ и $\beta(x,r)$ – пред- и постусловия рассматриваемого протокола [11].

Приведенные статические методы проверки непротиворечивости, полноты и целостности достаточны, т.е. доказывают отсутствие ошибок данного рода в проверяемой системе. С другой стороны, найденные недетерминизмы, тупики и нарушения условий целостности являются лишь подозрениями на наличие ошибок. Состояния среды, в которых проявляются найденные ошибки, можно получить из формул, построенных во время статического анализа. Достижимость этих состояний можно опровергать статическими методами, сформулировав соответствующие условия целостности, или доказывать с помощью генерации трасс и поиска в пространстве состояний [11].

4. Направленный поиск и создание тестовых сценариев. Реализован метод направленного поиска [8, 10], предназначенный для решения задачи генерации тестовых сценариев, удовлетворяющих заданным критериям покрытия. Такие критерии (гиды, Guides) формулируются в виде специальных регулярных выражений над алфавитом имен переходов модели. Гиды абстрактно определяют искомые поведения в терминах событий модели и одновременно накладывают ограничения на множество анализируемых состояний.

Семантически такие гиды задают «контрольные точки» поведения модели и определяют критерий (цепочку событий в поведении модели) выбора построенных трасс для последующего их использования в качестве тестовых сценариев [3, 4, 29, 30]. Так, предпо-

лагаемые поведения моделируемой системы проверяются на допустимость, при этом, по сути, осуществляется валидация модели и одновременно накладываются ограничения на поиск.

По каждому гиду строится соответствующий детерминированный конечный автомат. Реализованы точный и жадный алгоритмы распознавания для соответствующих типов регулярных выражений [8]. Важно отметить, что на практике удовлетворительный результат можно получить, используя жадные алгоритмы линейной сложности. Из соображений эффективности, реализация метода принимает на вход не один, а множество гидов, при этом осуществляется множественный поиск трасс одновременно всего набора гидов, а также вводятся ограничители на число необходимых трасс по каждому гиду. Алгоритм обхода пространства поведения модифицируется путем добавления перед шагом вглубь вызова специальной функции, реализующей автомат, распознающий гиды. Если текущая трасса удовлетворяет гиду, она будет сохранена на во внешней памяти, если не исчерпан соответствующий лимит (пользователь предварительно определяет нужное количество трасс по каждому гиду). Если лимит исчерпан, гид исключается из списка актуальных. Если данная функция возвращает на выходе $false$, это означает, что достигнута ситуация, из которой любое продолжение текущей трассы не приведет к обнаружению вхождения очередного символа (т.е. имени наблюдаемого перехода) хотя бы одного актуального гида. Такая ситуация является дополнительным критерием завершения генерации текущей трассы.

Метод позволяет эффективно находить труднодостижимые состояния, а так же проверять нарушение специфических условий безопасности —- например, наличие бесконечных циклов в поведении процесса при отсутствии внешних сигналов. Применение технологии направленного поиска в промышленных проектах дало возможность не только выявить ошибки в спецификациях, но и построить набор тестовых сценариев, обеспечивающих необходимое функциональное покрытие [4] при условии сохранения семантики требований.

Описанный программный комплекс осуществляет проверку всех наиболее распространенных свойств моделей, при этом выявленные случаи нарушения сопровождаются контрпримерами. Реализованы наиболее значимые оптимизации и методы редукции пространства состояний.

5. Описание UCM-нотации. Язык UCM [35] используется для описания поведения системы в виде совокупности взаимодействующих процессов. Конечный UCM-проект представляет собой набор связанных и структурированных диаграмм, каждая из которых состоит из последовательности элементов нотации [42], среди которых элементы описания событий, параллеизма, таймеров, прерываний и т.п. В совокупности набор диаграмм задает возможные поведения системы, описанные в требованиях. На рис. 1 приведена одна из UCM-диаграмм для телекоммуникационного протокола CDMA.

6. Расширение нотации. Нотация UCM позволяет в графическом виде задавать поведение системы с помощью явного представления потоков данных, но в то же время она не поддерживает отображение потоков данных и не позволяет детально описывать действия, производимые моделируемыми объектами. Для целей описания работы с потоком данных, точек отправки и приема сигналов и добавления условий ветвления и прерываний, нотация была расширена грамматикой метаданных. Информация из метаданных используется при формализации и полностью отображается в модели системы. На рис. 2 приведен пример использования метаданных для одного из элементов рассматриваемого проекта CDMA, в котором AC определяет действия, BP – базовые протоколы, SI – сигналы, VR предусловия и VO постусловия базового протокола. В метаданных есть возможность описания окружения системы, в которой определены начальные значения и атрибуты компонентов, без которых невозможна генерация покрывающего множества тестов в модели.

7. Инструментарий трансляции UCM в базовые протоколы. Для автоматического перехода от высокоуровневого дизайна, заданного в нотации UCM, к формальному описанию на языке базовых протоколов был разработан транслятор [20]. На рис. 3 приведена модульная структура транслятора. К основным компонентам транслятора можно отнести: модуль анализа UCM на наличие ошибок (1), в котором проверяется синтаксическая корректность метаданных и UCM конструкций; модуль обхода UCM-проекта (2), в котором реализована логика формирования скелета формальной модели системы; модуль оптимизации скелета модели (3), для повышения эффективности работы инструментов верификации и тестирования; модуль определения семантических кон-

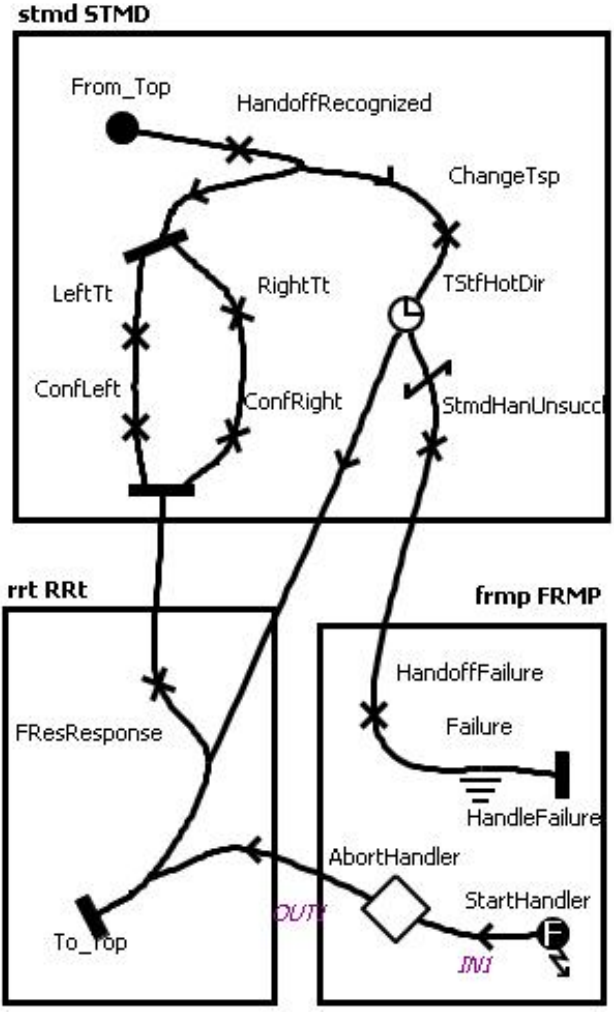

Рис. 1: UCM диаграмма для телекоммуникационного проекта.

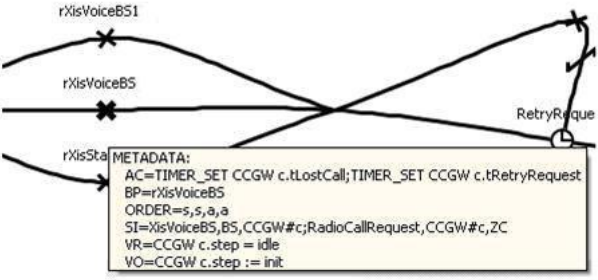

Рис. 2: Метаданные для элемента rXisVoiceBS.

струкций UCM (4), в котором происходит логическое связывание базовых протоколов в соответствии с семантикой UCM-элементов.

Транслятор поддерживает три различных способа формализации: метод «цветок», метод «дерево» и смешанный метод. Каждый из методов имеет свои преимущества и недостатки. В методе «цветок» предполагается, что очередность применения базовых протоколов базируется на значениях управляющей переменной агента.

Данный метод демонстрирует простоту в интерпретации с одной стороны, а с другой стороны ограничивает возможности работы со сложными моделями, использующими для описания поведения порядка 1000 базовых протоколов из-за «экспоненциального взрыва» числа анализируемых состояний. Важным модулем является модуль анализа UCM [19]. Разрабатывая модель, пользователь ориентируется на свое понимание требований к системе, свои знания и опыт при задании поведений. Но, несмотря на все усилия разработчика, модель системы может содержать ошибки. Анализатор на фазе проектирования предупреждает пользователя о потенциально опасных местах и помогает внести изменения в исходную модель. В методе «дерево» поток управления сохраняется за счет состояний агента, что дает значительное уменьшение времени на вычисление следующих за текущим узлом состояний по сравнению с методом «цветок». Указанное преимущество в несколько раз сокращает время анализа дерева поведения. Компромисс найден в смешанном методе, который наряду с уменьшением времени вычисления последовательностей базовых протоколов сохраняет простоту интерпретации.

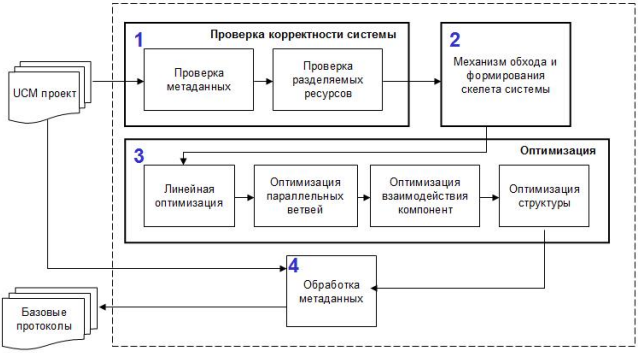

Рис. 3: Модульная структура транслятора UCM проекта в базовые протоколы.

Сценарий использования инструмента. После разработки UCM-проекта происходит проверка наличия ошибок с помощью анализатора. Если ошибки найдены, то пользователю требуется внести изменения в исходные UCM-диаграммы. Затем, осуществляется настройка опций транслятора. Выбор настроек основан на целях использования формальной модели. После трансляции полученный набор базовых протоколов импортируется в верификатор, в котором происходит верификация модели и генерация тестов. Полученный набор тестов используется в системе автоматизации тестирования.

Метод автоматизированного создания тестового набора. Для осуществления проверки системы на соответствие требованиям необходимо обеспечить контролируемый процесс запуска тестов и анализа их результатов [31], что решается через создание тестового набора, который бы позволял описывать взаимодействие тестируемой системы с окружением в том виде, в котором это соответствует спецификации. Наблюдение поведения системы при выполнении данного тестового набора позволяет вынести вердикт, экспериментально подтверждающий корректность реализации, либо опровергающий ее [43].

Для построения тестового набора предлагается автоматизированный подход, основанный на использовании инструментария трансляции UCM в базовые протоколы. Общая схема подхода

представлена на рис. 4. Являясь объединением описанной техники формализации требований и технологии генерации поведенческих трасс на основании модели базовых протоколов [49], данный подход позволяет объединить в единую технологическую цепочку управление требованиями, верификацию и тестирование. При этом автоматически созданные поведенческие сценарии содержат описание взаимодействий всех сущностей системы и её окружения, что позволяет на этапе тестирования проверить все зафиксированные в требованиях взаимодействия.

Данный метод обеспечивает автоматизацию всех рутинных этапов работы. Нотации, используемые в качестве формата тестовых наборов (UCM, MSC [41]), просты в понимании человеком, а также позволяют использовать методы верификации для проверки требований [44].

Система автоматического тестирования. Для автоматического исполнения тестового набора и анализа результатов создан инструментарий TestCommander. В общем виде создаваемый им тестовый набор и его взаимодействие с тестируемой системой (SUT — System Under Test) представлены на рис. 4.

В TestCommander допускается использование любого способа создания диаграмм MSC, как вручную, так и с использованием автоматизированного подхода, основанного на использовании инструментария трансляции UCM в базовые протоколы.

Генерируемый системой тестирования тестовый набор состоит из одного или нескольких тестирующих и одного управляющего модулей. Тестирующий модуль взаимодействует с SUT согласно логике теста и обменивается контрольными сигналами с управляющим модулем. Он централизованно руководит процессом тестирования и сбором результатов. Для определения протокола взаимодействия между элементами тестирующей системы, а также самой тестирующей системы с окружением, используется язык спецификации протоколов (PSL — Protocol specification language). Он позволяет однозначно определить вид сообщений, которыми обмениваются участвующие в тестировании сущности. В рамках рассматриваемого в данной работе подхода предполагается ручное создание спецификации на языке PSL.

Для конфигурирования тестового набора, задания параметров генерации кода и развёртывания процесса тестирования применяется конфигурационный файл в формате JSON [49]. Он позволя-

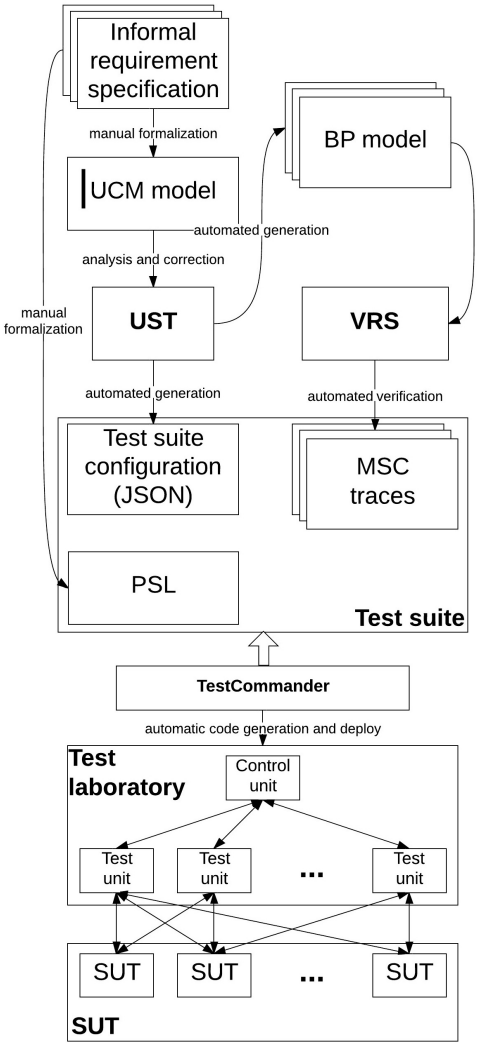

Рис. 4: Общая схема подхода к распределённому тестированию.

ет определить физическое расположение модулей тестового набора и SUT, а также сконфигурировать тестовый набор, заменив часть элементов тестируемой системы тестирующими модулями. При этом проверяется поведение не всей системы, а только отдельных её компонентов. Данный файл создаётся автоматически на основании представления системы в UCM, и на этапе создания тестового набора пользователь может вручную задать необходимые значения.

После определения параметров тестового набора (участвующие в тестировании элементы системы, физическое расположение тестового набора и другие) автоматически создаётся код тестового набора на целевом языке и происходит развёртывание (загрузка на целевой компьютер). Запуск теста осуществляется путём старта процесса управляющего модуля тестового набора, который самостоятельно осуществляет запуск и остановку тестирующих модулей и SUT. Результаты формируются в виде журнала событий и MSC-диаграмм хода тестирования.

Рассмотренный метод объединяет в себе элементы управления требованиями, верификации и тестирования и позволяет осуществлять проверку системы на соответствие спецификации в едином производственном цикле. Тестирование осуществляется на основании набора тестовых сценариев, корректность которых доказана при верификации модели системы. При этом существенно снижаются затраты, связанные с регрессионным тестированием при изменении требований или их дополнении. Этапы данного технологического процесса частично или полностью автоматизированы. Программные инструменты, относящиеся к различным этапам работы, взаимонезависимы; использующиеся форматы представления данных стандартизованы и имеют точную спецификацию. Описанный набор инструментов в рамках разработанной технологии позволяет обеспечить свойства масштабируемости, адаптируемости под условия конкретной задачи и открытости для модернизации.

8. Верификатор SDL-спецификаций распределенных систем. Разработана и реализована новая версия программного комплекса SRDSV2 (SDL/REAL Distributed Systems Verifier), предназначенного для моделирования, анализа и верификации SDL-спецификаций распределенных систем, который использует разработанный ранее язык Dynamic-REAL (dREAL) в качестве промежуточного языка [16–18].

Язык dREAL расширяет язык Basic-REAL [46] посредством динамических конструкций порождения и уничтожения процессов. Язык dREAL состоит из языка выполнимых спецификаций для представления распределённых систем и языка логических спецификаций для представления их свойств. Выполнимая спецификация на языке dREAL имеет иерархическую структуру и состоит из заголовка, контекста, диаграммы и подспецификаций. Заголовок определяет имя и вид спецификации: блок или процесс. Контекст состоит из определений типов и описаний переменных и каналов. Диаграмма блока состоит из маршрутов, соединяющих подблоки между собой и с внешней средой. Диаграмма процесса состоит из переходов, которые включают имя состояния, тело, временной интервал и список (возможно, пустой) следующих состояний. Имя состояния служит меткой, причём одно состояние может метить несколько переходов. Каждый экземпляр процесса может порождать или уничтожать экземпляры процессов в своем блоке. Подобно языку SDL, в dREAL количество экземпляров, существующих с самого начала, и максимально возможное количество экземпляров определяется в заголовке процесса. Версия языка dREAL, подробно описанная в [18], расширена за счёт добавления макроопределений процедур, не содержащих обращений к каналам. Язык логических спецификаций расширяет конструкции логики ветвящегося времени CTL (Computational Tree Logic) за счет использования временных интервалов и средств динамических логик. Формулы этого языка строятся из предикатов с помощью булевских операций, модальностей по моментам времени («всегда» или «иногда») и по возможным поведениям выполнимых спецификаций («для всех поведений» или «для некоторых поведений»), а также кванторов (существования и всеобщности) по выделенным переменным и по экземплярам процессов.

Программный комплекс SRDSV2 включает транслятор из SDL в dREAL, системы симуляции и автоматического моделирования dREAL-спецификаций, а также использует известную систему верификации методом проверки моделей SPIN. Входной язык этого транслятора включает все базовые конструкции языка SDL (кроме абстрактных типов данных), которые образуют язык SDL-88 и широко применяются на практике. Система автоматического моделирования, которая использует запросы пользователя, играет важную роль, так как может успешно применяться и в тех случаях,

когда система верификации SPIN неприменима из-за громоздких моделей SDL-спецификаций. Описанное выше расширение языка dREAL позволило в ряде случаев оптимизировать модели dREAL-спецификаций и применить к их верификации систему SPIN. Были проведены успешные эксперименты по автоматическому моделированию и верификации новой версии динамической системы управления сетью касс-терминалов.

9. Верификатор автоматных спецификаций телекоммуникационных систем. Были определены униформные системы взаимодействующих расширенных конечных автоматов (РКА), которые удобны для задания исходной спецификации телекоммуникационных систем, таких как кольцевые протоколы и телефонные сети [5, 32]. Система взаимодействующих РКА — это набор автоматов, которые изменяют значения своих локальных и глобальных переменных и взаимодействуют друг с другом, отправляя сигналы. Сигналы могут иметь разные типы, включающие идентификаторы отправителя и получателя, а также некоторую дополнительную информацию. Такая система работает пошагово. На каждом шагу может сработать некоторый переход автомата, если этот переход разрешён, т.е. все условия его срабатывания выполнены (локальные и глобальные переменные имеют определённые значения, и есть входной сигнал определённого типа). Когда переход срабатывает, автомат меняет своё состояние, удаляет соответствующий входной сигнал из окружения, выполняет некоторые вычисления на локальных и глобальных переменных и посылает выходные сигналы в окружение. Был разработан и формально обоснован алгоритм трансляции таких систем РКА в раскрашенные сети Петри (CPN — Colored Petri Net) [32].

Была разработана и реализована новая версия программного комплекса ASV (Automata Systems Verifier), предназначенного для анализа и верификации автоматных спецификаций телекоммуникационных систем. Он базируется на алгоритме трансляции систем РКА в CPN и применяет для анализа автоматных спецификаций различные средства анализа CPN, реализованные в известной системе CPN Tools. Комплекс ASV включает впервые разработанный транслятор из CPN во входной язык Promela известной системы верификации SPIN, которая применяет метод проверки моделей относительно свойств, заданных формулами логики линейного времени LTL. Были проведены новые эксперименты по применению

комплекса ASV для верификации взаимодействия функциональностей в телефонных сетях.

10. Система верификации Спектр-К. Система Спектр-К представляет собой компонент мультиязыковой системы анализа и верификации программ Спектр [13,26], ориентированный на верификацию телекоммуникационных систем, представленных на языке C. Особенность системы Спектр заключается в том, что она использует предметно-ориентированный язык выполнимых спецификаций Atoment [2, 27], предназначенный для спецификации методов и средств верификации программ. Таким образом, система Спектр-К определяется как набор спецификаций на языке Atoment.

Система Спектр-К базируется на дедуктивном подходе к верификации телекоммуникационных систем, представленных на языке C [1]. Этот подход сводит верификацию параллельных взаимодействующих компонентов телекоммуникационной системы, представленных на языке C, к верификации последовательных изолированных компонентов, представленных на языке Atoment. Сведение состоит в явном определении взаимодействующих процессов телекоммуникационной системы (в частности, окружения), выделении и спецификации состояний этих процессов, замене взаимодействий процессов их обращением к разделяемым переменным, расстановки означиваний событийных переменных, которые используются для идентификации состояний процессов, и расстановки контрактов безопасности (инвариантов) для разделяемых переменных. Подход применяется к представительному подмножеству языка C — языку C-light [14, 45], имеющему формальную операционную семантику, и использует двухуровневую схему верификации C-light-программ, при которой C-light-программы сначала транслируются с сохранением аннотаций в программы на промежуточном языке C-kernel, а затем для этих программ порождаются условия корректности на основе аксиоматической семантики языка C-kernel [15].

Технологической составляющей дедуктивного подхода к верификации телекоммуникационных систем являются специализированные помеченные системы переходов [28], ориентированные на разработку средств спецификации и верификации программных систем. Подход использует два класса таких систем — ориентированные на операционную семантику системы переходов и ори-

ентированные на логику безопасности системы переходов. Первый класс систем предназначен для разработки операционной семантики программных систем (в частности, языков программирования). Второй класс специфицирует правила новой логики программ — логики безопасности. Логика безопасности является расширением логики Хоара, ориентированным на доказательство свойств безопасности программ. В частности, она позволяет проверять свойства безопасности незавершающихся программ в отличие от логики Хоара, ориентированной на доказательство корректности завершающихся программ. Поэтому, логика безопасности лучше подходит для доказательства свойств безопасности телекоммуникационных систем по сравнению с логикой Хоара, поскольку эти системы описываются, как правило, как системы бесконечных взаимодействующих процессов. В то же время, логика безопасности сохраняет такое преимущество дедуктивной верификации как применимость к системам с бесконечным (или очень большим) числом состояний в отличие от метода проверки моделей.

Система Спектр-К включает транслятор C-light-программ в выражения языка Atoment, ориентированную на операционную семантику систему переходов, специфицирующую алгоритм сведение паралельных процессов к последовательным; ориентированную на логику безопасности систему переходов для C-kernel-программ с разделяемыми переменными. В качестве решателей для порождаемых логикой безопасности условий корректности используются SMT-решатели Z3 и Simplifier.

В настоящее время система Спектр-К используется для проверки свойств безопасности протоколов передачи данных в телекоммуникационных протоколах [25]. Поэтому, дополнительно к выше перечисленным компонентам, она включает правила логики безопасности для функций стандартной библиотеки языка C, используемых в этих протоколах, и правила логики безопасности для C-функций из специализированной библиотеки, описывающей взаимодействие параллельных процессов. Формальная модель (операционная семантика) такого взаимодействия строится на базе ориентированных на операционную семантику систем переходов.

11. Заключение. В работе рассмотрен набор методов и инструментальных средств, решающих задачу интегрированной верификации и тестирования промышленных программных проектов. Главным достоинством разработанного подхода является ре-

шение задачи ограничения взрыва состояний, характерной для промышленных проектов среднего и большого размера.

В работе описаны интегрированная технология и прототипы новых программных средств для автоматической верификации программных систем, которые были реализованы и проверены в рамках совместной работы указанных трех коллективов над грантом РФФИ 11-07-90412-Укр_ф_а. В сочетании с новыми реализованными возможностями языков SDL, UCM, dREAL и C/C++ для задания основных поведенческих режимов рассматриваемых телекоммуникационных приложений данная технология позволяет порождать из проверенных спецификаций тестовые наборы, обеспечивающие заданные критерии покрытия исходных спецификаций.

Следует отметить, что в качестве средства формализации требований на поведение приложений наряду с традиционными языками используются UCM-диаграммы, позволяющие представить реализацию требований в нотации, доступной для контроля заказчиком, что позволяет сохранить соответствие семантики генерируемых поведенческих моделей семантике исходных требований.

Данная технология и поддерживающие инструментальные средства открывают новые возможности для массовой промышленной разработки программных продуктов высокого качества.

Список литературы

[1] *Анурeев И.С.* Дедуктивная верификация телекоммуникационных систем, представленных на языке Си // Моделирование и анализ информационных систем. 2012. Т. 19, вып. 4. 10 с. (в печати)

[2] *Анурeев И.С.* Типовые примеры использования языка Atoment // Моделирование и анализ информационных систем. 2011. Т. 18, вып. 4. С. 7–20.

[3] *Баранов С.Н., Котляров В.П.* Формальная модель требований, используемая в процессе генерации кода приложения и кода тестов // Моделирование и анализ информационных систем. 2011. Т. 18, вып. 4. С. 118–130.

[4] *Баранов С.Н., Котляров В.П., Летичевский А.А.* Индустриальная технология автоматизации тестирования мобильных

устройств на основе верифицированных поведенческих моделей проектных спецификаций требований // Труды междунар. науч. конф. «Космос, астрономия и программирование» СПб: СПбГУ, 2008. С. 134–145.

[5] *Белоглазов Д.М., Машуков М.Ю., Непомнящий В.А.* Верификация телекоммуникационных систем, специфицированных взаимодействующими конечными автоматами, с помощью раскрашенных сетей Петри // Моделирование и анализ информационных систем. 2011. Т. 18, вып. 4. С. 144–156.

[6] *Дробинцев П.Д., Колчин А.В., Котляров В.П., Летичевский А.А., Песчаненко В.С.* Подход к конкретизации тестовых сценариев в рамках технологии автоматизации тестирования промышленных программных проектов // Моделирование и анализ информационных систем. 2012. Вып. 4.(в печати)

[7] *Колчин А.В.* Автоматический метод динамического построения абстракций состояний формальной модели // Кибернетика и системный анализ. 2010. Вып. 4. С. 70–90.

[8] *Колчин А.В.* Метод направления поиска и генерации тестовых сценариев при верификации формальных моделей асинхронных систем // Проблемы программирования. 2008. Вып. 4. С. 5–12.

[9] *Колчин А.В.* Оптимизация проверки выполнимости переходов при верификации формальных моделей // Проблемы программирования. 2012. Вып. 2–3. С. 201–210.

[10] *Колчин А.В., Котляров В.П., Дробинцев П.Д.* Метод генерации тестовых сценариев в среде инсерционного моделирования // Управляющие системы и машины. 2012. Вып. 6. 9 с. (в печати)

[11] *Летичевский А.А., Годлевский А.Б., Летичевский А.А.(мл.), Потиенко С.В., Песчаненко В.С.* Свойства предикатного трансформера системы VRS // Кибернетика и системный анализ. 2010. Вып. 4. С. 3–16.

[12] *Летичевский А.А., Капитонова Ю.В., Волков В.А., и др.* Спецификация систем с помощью базовых протоколов // Кибернетика и Системный Анализ. 2005. Вып. 4. С. 3–21.

[13] *Непомнящий В.А., Анureев И.С., Атучин М.М., Марьясов И.В., Петров А.А., Промский А.В.* Верификация С-программ в мультиязыковой системе СПЕКТР // Моделирование и анализ информационных систем. 2010. Т. 17, вып. 4. С. 88–100.

[14] *Непомнящий В.А., Анureев И.С., Михайлов И.Н., Промский А.В.* На пути к верификации С программ. Язык C-light и его формальная семантика //Программирование. 2002. Вып. 6. С. 1–13.

[15] *Непомнящий В.А., Анureев И.С., Промский А.В.* На пути к верификации С программ. Аксиоматическая семантика языка C-kernel // Программирование. 2003. Вып. 6. С. 65–80.

[16] *Непомнящий В.А., Бодин Е.В., Веретнов С.О.* Моделирование и верификация распределенных систем, представленных на языке SDL, с помощью языка Dynamic-REAL. Препринт ИСИ СО РАН. 2010. № 156. 44 с.

[17] *Непомнящий В.А., Бодин Е.В., Веретнов С.О.* Применение языка Dynamic-REAL для анализа и верификации распределенных систем, специфицированных на языке SDL. Препринт ИСИ СО РАН. 2011. № 161. 52 с.

[18] *Непомнящий В.А., Бодин Е.В., Веретнов С.О.* Язык спецификаций распределенных систем Dynamic-REAL. Препринт ИСИ СО РАН. 2007. № 147. 42 с.

[19] *Никифоров И.В., Васин А.А., Котляров В.П.* Анализ зависимостей по данным в UCM проекте // Технологии Microsoft в теории и практике программирования. Материалы межвузовского конкурса-конференции студентов, аспирантов и молодых ученых Северо-Запада 27 марта 2012 г. 2012. СПб: Изд-во Политехнического ун-та. С. 71–72.

[20] *Никифоров И.В., Петров А.В., Юсупов Ю.В.* Генерация формальной модели системы по требованиям, заданным в нотации USE CASE MAPS // Научно-технические ведомости СПбГПУ. Вып. 4(103). СПб: Изд-во Политехнического ун-та, 2010. С. 191–195.

[21] *Потиенко С.В.* Организация базы знаний о переходах системы с атрибутами перечислимых типов // Управляющие системы и машины. 2012. Вып. 6. 9 с. (в печати)

[22] *Потиенко С.В.* Статическая проверка требований и подходы к решению проблемы достижимости // Искусственный интеллект. 2009. Вып. 1. С. 192–197.

[23] *Потиенко С.В., Колчин А.В.* Представление SDL-спецификаций в виде базовых протоколов // Искусственный интеллект. 2006. Вып. 4. С. 42–52.

[24] *Потиенко С.В., Колчин А.В.* Трансляция MSC сценариев в язык базовых протоколов // Искусственный интеллект. 2007. Вып. 3. С. 428–435.

[25] *Таненбаум Э.* Компьютерные сети. 4-е издание, 2003. 992 с.

[26] *Anureev I.S.* Integrated approach to analysis and verification of imperative programs // Joint NCC&IIS Bulletin (Series Computer Science). 2011. Vol. 32. P. 1–18.

[27] *Anureev I.S.* Introduction to the Atoment language // Joint NCC&IIS Bulletin (Series Computer Science). 2010. Vol. 31. P. 1–16.

[28] *Anureev I.S.* Program specific transition systems // Joint NCC&IIS Bulletin (Series Computer Science). 2012. Vol. 32. 21 p. (to appear)

[29] *Baranov S., Kotlyarov V.* A Formal Application Model for Code and Test Generation // Automatic Control and Computer Sciences. 2012. Vol. 46, iss. 7. Allerton Press, Inc. P. 371–378.

[30] *Baranov S., Kotlyarov V., Weigert T.* Varifiable Coverage Criteria For Automated Tesdting // SDL2011: Integrating System and Software Modeling. (Lecture Notes in Computer Science). 2012. Vol. 7083. P. 79–89.

[31] *Beck K.* Test-Driven Development by Example. Saint-Petersburg: Piter, 2003.

[32] *Beloglazov D., Nepomniaschy V.* A Two-Level Approach for Modeling and Verification of Telecommunication Systems // PSI-2003. Proc. of Conf. (Lect. Notes Comput. Sci.). 2010. Vol. 5947. P. 70–85.

[33] *Ben-Ari M.*Principles of Spin. Springer Verlag, 2008. 216 p.

[34] *Beyer D., Henzinger T., Jhala R., Majumdar R.* The software model checker BLAST // Int J Softw Tools Technol Transfer. 2007. Iss. 9. P. 505–525.

[35] *Buhr R.J.A. and Casselman R.S.* Use Case Maps for Object-Oriented Systems. London: Prentice Hall, 1996.

[36] CBMC – Bounded Model Checking for ANSI-C.(electronic) http://www.cs.cmu.edu/ modelcheck/cbmc

[37] *Cimatti A., Clarke E. M., Giunchiglia E., and others* NuSMV 2: An OpenSource Tool for Symbolic Model Checking // Proceeding of International Conf. on Computer-Aided Verification. Copenhagen, 2002. P. 359–364.

[38] *Godefroid P.* Partial-order methods for the verification of concurrent systems – an approach to the state-explosion problem // Lecture notes in computer science. Springer-Verlag, 1996. Vol. 1032. P. 143.

[39] *Godefroid P.* Software model checking: the VeriSoft approach // Formal methods in system design. Springer science, 2005. Vol. 26. P. 77–101.

[40] *Ip C., Dill D.* Verifying Systems with Replicated Components in Murphi // International Conf. on Computer-Aided Verification. 1996. P. 147–158.

[41] ITU-T Recommendation Z.120: Message sequence chart (MSC) (Geneva, Switzerland, October, 1996) (electronic) http://eu.sabotage.org/www/ITU/Z/Z0120e.pdf

[42] ITU-T Recommendation Z.151 : User requirements notation (URN) - Language definition (Geneva, Switzerland, September, 2003) (electronic) http://www.itu.int/rec/T-REC-Z.151-200811-I/en

[43] *Kaner C., Falk J., Nguyen H. Q.* Testing Computer Software (2nd ed.). New York: Wiley, 1999.

[44] *Letichevsky A. et al.* Basic protocols, message sequence charts, and the verification of requirements specifications // Computer Networks: The International Journal of Computer and Telecommunications Networking. 2005. Vol. 49, iss. 5. P. 661–675.

[45] *Nepomniaschy V.A., Anureev I.S., Promsky A.V.* Verification-oriented language C-light and its structural operational semantics // PSI-2003. Proc. of Conf. (Lect. Notes Comput. Sci.). 2003. Vol. 2890. P. 1–5.

[46] *Nepomniaschy V.A., Shilov N.V., Bodin E.V., Kozura V.E.* Basic-REAL: integrated approach for design, specification and verification of distributed systems // Proc. of IFM 2002. (Lect. Notes Comput. Sci.). 2002. Vol. 2335. P. 69–88.

[47] *Peled D.* Combining partial order reductions with on-the-fly model checking // Journal of Formal Methods in System Design. 1996. Vol. 8, iss. 1. P. 39–64.

[48] VCEGAR – Verilog Counterexample Guided Abstraction and Refinement (electronic) http://www.cs.cmu.edu/ modelcheck/vcegar

[49] *Veselov A.O., Kotlyarov V. P.* Testing automation of projects in telecommunication domain // Proceedings of the 4th Spring/Summer Young Researchers' Colloquium on Software Engineering (Nizhny Novgorod, June 1-2, 2010) P. 81–86.

Ануреев Игорь Сергеевич — к.ф.-м.н.; с.н.с. лаборатории теоретического программирования Института систем информатики имени А.П. Ершова СО РАН (ИСИ СО РАН). Область научных интересов: компьютерные языки, верификация программных систем, спецификация программных систем, семантика программных систем, автоматическое доказательство, безопасность программных систем, онтологические модели, семантика компьютерных языков, предметно-ориентированные языки. Число научных публикаций — 70. anureev@iis.nsk.su; ИСИ СО РАН, пр. Академика Лаврентьева, д. 6, г. Новосибирск, 630090, РФ; р.т. +7(383)330-6360, факс +7(383)332-3494.

Igor S. Anureev — PhD in Computer Science; senior researcher of Theoretical Programming Laboratory of A.P. Ershov Institute of Informatics Systems, Siberian Branch of the Russian Academy of Sciences (IIS SB RAS). Research area: computer

languages, verification of program systems, specification of program systems, semantics of program systems, automated proving, safety of program systems, ontological models, semantics of computer languages, domain-specific languages. Number of publications — 70. anureev@iis.nsk.su; IIS SB RAS, 6, Acad. Lavrentjev, Novosibirsk 630090, Russia; office phone +7(383)330-6360, fax +7(383)332-3494.

Баранов Сергей Николаевич — д.ф.-м.н., проф.; гл.н.с. лаборатории технологий и систем программирования СПИИРАН. Область научных интересов: технология программирования, промышленная разработка программных изделий, формальные методы верификации требований, формальные языки. Число научных публикаций — 90+. SNBaranov@gmail.com; СПИИРАН, 14-я линия В.О., д. 39, г. Санкт-Петербург, 199178, РФ; р.т./факс +7(812)328-0887.

Sergey N. Baranov — Doc.Nat.Sci, Professor; Chief Research Associate, lab of software engineering and systems, SPIIRAS. Research area: software engineering, industrial development of software products, formal methods for requirements verification, formal languages. Number of publications — 90+. SNBaranov@gmail.com; SPIIRAS, 14-th line V.O., 39, St. Petersburg, 199178, Russia; office phone/fax +7(812)328-0887.

Белоглазов Дмитрий Михайлович — м.н.с. лаборатории теоретического программирования Института систем информатики имени А.П. Ершова СО РАН (ИСИ СО РАН). Область научных интересов: верификация, спецификация, коммуникационные протоколы, сети Петри, проверка моделей, телекоммуникационные системы, телефонные сети. Число научных публикаций — 11. dmitry.beloglazov@gmail.com; ИСИ СО РАН, пр. Академика Лаврентьева, д. 6, г. Новосибирск, 630090, РФ; р.т. +7(383)330-6360, факс +7(383)332-3494.

Dmitry M. Beloglazov — Junior researcher of Theoretical Programming Laboratory of A.P. Ershov Institute of Informatics Systems, Siberian Branch of the Russian Academy of Sciences (IIS SB RAS). Research area: verification, specification, communication protocols, Petri nets, model checking, telecommunication systems, telephone networks. Number of publications — 11. dmitry.beloglazov@gmail.com; IIS SB RAS, 6, Acad. Lavrentjev pr., Novosibirsk 630090, Russia; office phone +7(383)330-6360, fax +7(383)332-3494.

Бодин Евгений Викторович — н.с. лаборатории теоретического программирования Института систем информатики имени А.П. Ершова СО РАН (ИСИ СО РАН). Область научных интересов: коммуникационные протоколы, спецификация, верификация, трансляция, семантика, язык SDL, язык REAL, системы верификации, метод проверки моделей, телекоммуникационные системы. Число научных публикаций — 22. bodin@iis.nsk.su; ИСИ СО РАН, пр. Академика Лаврентьева, д. 6, г. Новосибирск, 630090, РФ; р.т. +7(383)330-6253, факс +7(383)332-3494.

Evgeny V. Bodin — researcher of Theoretical Programming Laboratory of A.P. Ershov Institute of Informatics Systems, Siberian Branch of the Russian Academy of Sciences (IIS SB RAS). Research area: specification, verification, communication protocols, translation, semantics, telecommunication systems, language REAL, language SDL, verification systems, model checking method. Number of publications — 22. bodin@iis.nsk.su; IIS SB RAS, 6, Acad. Lavrentjev pr., Novosibirsk 630090, Russia; office phone +7(383) 330-6253, fax +7(383)332-3494.

Дробинцев Павел Дмитриевич — к.т.н., доцент кафедры информационных и управляющих систем Санкт-Петербургского государственного политехнического университета. Область научных интересов: формальные методы, верификация программного обеспечения, спецификации, тестирование телекоммуникационных систем; автоматизация индустриальной разработки программного продукта. Число научных публикаций — 31. drob@ics2.ecd.spbstu.ru; СПбГПУ, ул. Политехническая, д.29, Санкт-Петербург, 195251, РФ; р.т. +78122971628.

Pavel D. Drobintsev — PhD, Associate Professor of Information and Control System Chair of St.Petersburg State Politechnic University. Research area: formal methods, verification of software systems, software specification, testing of telecom systems, automation of industrial technologies for software development. Number of publications — 31. drob@ics2.ecd.spbstu.ru; SPBGPU, Politechnicheskaya st. 29. St.Petersburg, 195251, Russia. work phone: +78122971628.

Колчин Александр Валентинович — к.ф.-м.н., н.с. отдела теории цифровых автоматов Института кибернетики НАН Украины имени В.М. Глушкова. Область научных интересов: верификация, проверка моделей. Число научных публикаций - 24. kolchin_av@yahoo.com; Институт кибернетики НАНУ имени В.М. Глушкова, пр. Академика Глушкова, 40, г. Киев, 03680 МСП, Украина; тел. +38(044)526-0058, факс +38(044)526-7418.

Alexander V. Kolchin — PhD, research associate of Dep. of Theory of Digital Automata, V.M.Glushkov Institute of Cybernetics, National Ukrainian Academy of Sciences. Research area: verification, model checking. Number of publications - 24. kolchin_av@yahoo.com; Glushkov Institute of Cybernetics, Acad. Glushkov pr. 40, Kiev 03680 MSP, Ukraine; office phone +38(044)526-0058, fax +38(044)526-7418.

Котляров Всеволод Павлович — профессор кафедры информационных и управляющих систем Санкт-Петербургского государственного политехнического университета. Область научных интересов: программная инженерия, инженерия требований, формальные методы, верификация программного обеспечения, спецификации, тестирование телекоммуникационных систем; автоматизация индустриальных технологий разработки программного продукта. Число научных публикаций — 240. vpk@spbstu.ru; СПбГПУ, ул. Политехническая, д.29, г. Санкт-Петербург, 195251, РФ; р.т. +7(812)552-7666.

Vsevolod P. Kotlyarov — Professor of Information and Control System Chair of St.Petersburg State Politechnic University. Research area: software engineering, requirement engineering, formal methods, verification of software systems, software specification, testing of telecom systems, automation of industrial technologies for software development. Number of publications — 240. vpk@spbstu.ru; SPBGPU, Politechnicheskaya, 29. St.Petersburg, 195251, Russia; office phone: +78122971628.

Летичевский Александр Адольфович — академик НАН Украины, зав. отделом теории цифровых автоматов Института кибернетики имени В.М. Глушкова НАН Украины. Область научных интересов: символьная верификация требований, доказательное моделирование, алгебраическое и инсерционное программирование. Число научных публикаций - более 200. let@cyfra.net ул. Пирогова 2, кВ.113 г. Киев, Украина; тел. +38(044)526-0058, факс +38(044)526-7418.

Alexander A. Letichevsky — Academician, Chief of department of Theory of Digital Automata of V.M.Glushkov Institute of cybernetics, National Ukrainian

Academy of Sciences. Research area: symbolic verification of requirements, proving modeling, algebraic and insertion programming. Number of publications - 200. let@cyfra.net; 113, 2 Pirogova st., Kiev, Ukraine; office phone +38(044)526-0058, fax +38(044)526-7418.

Летичевский Александр Александрович — к.ф.-м.н., с.н.с. отдела теории цифровых автоматов Института кибернетики имени В.М. Глушкова НАН Украины. Область научных интересов: символьная верификация требований, доказательное моделирование, дедуктивное тестирование. Число научных публикаций - 42.; lit@iss.org.ua проспект 40 лет Октября 82, 76, Киев, 03040, Украина; тел. +38(044)526-0058.

Oleksandr A. Letychevskyi — PhD, senior researcher of department of Theory of Digital Automata of V.M.Glushkov Institute of Cybernetic, National Ukrainian Academy of Sciences. Research area: symbolic verification of requirements, proving modeling, deductive testing. Number of publications - 42. lit@iss.org.ua ; 76, 82, 40 richya Zhovtnya pr., Kiev, 03040, Ukraine; office phone +38(044)526-0058.

Непомнящий Валерий Александрович — к.ф.-м.н.; зав. лабораторией теоретического программирования Института систем информатики имени А. П. Ершова СО РАН (ИСИ СО РАН). Область научных интересов: теория программирования, верификация, спецификация, семантика, системы верификации программ, коммуникационные протоколы, сети Петри, телекоммуникационные системы. Число научных публикаций — 172. vnep@iis.nsk.su; ИСИ СО РАН, пр. Академика Лаврентьева, д. 6, г. Новосибирск, 630090, РФ; р.т. +7(383)330-6360, факс +7(383)332-3494.

Valery A. Nepomniaschy — PhD; head of Theoretical Programming Laboratory of A.P. Ershov Institute of Informatics Systems, Siberian Branch of the Russian Academy of Sciences (IIS SB RAS). Research area:programming theory, verification, specification, semantics, communication protocols, verification program systems, Petri nets, telecommunication systems. Number of publications — 172. vnep@iis.nsk.su; IIS SB RAS, 6, Acad. Lavrentjev pr., Novosibirsk 630090, Russia; office phone +7(383)330-6360, fax +7(383)332-3494.

Никифоров Игорь Валерьевич — аспирант кафедры информационных и управляющих систем Санкт-Петербургского государственного политехнического университета (СПбГПУ). Область научных интересов: методы формализации, спецификации, тестирования телекоммуникационные систем; автоматизация индустриальных технологий разработки программного продукта. Число научных публикаций — 17. igor.nikiforovv@gmail.com; СПбГПУ, ул. Политехническая, д.29, г. С.Петербург, 195251, РФ; р.т. +7(812)552-7666.

Igor V. Nikiforov — PhD student, Dep. of Technical Cybernetics, Chair of Information and Control Systems, St.Petersburg State Polytechnic University. Research area: formalization methods, specification, testing of telecommunication systems, model checking, automatization of industrial software development. Number of publications — 17. igor.nikiforovv@gmail.com; Saint-Petersburg, Politeknicheskaya, 29, 195251, Russia; office phone: +7(812)552-7666.

Потиенко Степан Валериевич — к.ф.-м.н., н.с. отдела теории цифровых автоматов Института кибернетики имени В.М. Глушкова НАН Украины. Область научных интересов: верификация, формальные методы, символьное моделирование, проверка моделей. Число научных публикаций - 15.

stepan@iss.org.ua; Институт кибернетики имени В.М. Глушкова НАНУ, пр. Академика Глушкова, 40, Киев, 03680 МСП, Украина; тел. +38(044)526-0058, факс +38(044)526-7418.

Stepan V. Potiyenko — PhD, researcher of department of Theory of Digital Automata of V.M.Glushkov Institute of cybernetics, National Ukrainian Academy of Sciences. Research area: verification, formal methods, symbolic modeling, model checking. Number of publications - 15. stepan@iss.org.ua; V.M. Glushkov Institute of cybernetics, Acad. Glushkov pr. 40, Kiev 03680 MSP, Ukraine; office phone +38(044)526-0058, fax +38(044)526-7418.

Прийма Лев Викторович — аспирант факультета технической кибернетики Санкт-Петербургского государственного политехнического университета. Область научных интересов: Методы обеспечения качества программных продуктов, тестирование, верификация, спецификация, проверка моделей, статический и динамический анализ кода. Число научных публикаций — 3. levpri@gmail.com; ФГБОУ ВПО "СПбГПУ ул. Политехническая, д. 29, Санкт-Петербург, 195251, РФ; р.т. +7(812)552-7666, факс +7(812) 552-6080.

Lev V. Priima — PhD student of department of Technical Cybernetics of St. Petersburg State Polytechnical University (SPBSPU). Research area: Software quality assurance, testing, verification, specification, model checking, static and dynamic code analysis. Number of publications — 3. levpri@gmail.com; SPBSPU, 29, Polytechnicheskaya st., St.Petersburg 195251, Russia; office phone +7(812)552-7666, fax+7(812) 552-6080.

Тютин Борис Викторович — аспирант факультета технической кибернетики Санкт-Петербургского государственного политехнического университета. Область научных интересов: теория алгоритмов, программная инженерия, формальные нотации, тестирование, верификация. Число научных публикаций — 17. b.tyutin@ics2.ftk.spbstu.ru; ФГБОУ ВПО "СПбГПУ ул. Политехническая, д.29, Санкт-Петербург, 195251, РФ; р.т. +7(812)552-7666, факс +7(812) 552-6080.

Boris V. Tyutin — PhD student of department of Technical Cybernetics of St. Petersburg State Polytechnical University. Research area: algorythms, program engineering, formal notations, testing, verification. Number of publications — 17. b.tyutin@ics2.ftk.spbstu.ru; SPBSPU, 29, Polytechnicheskaya, St.Petersburg, 195251, Russia; office phone +7(812)552-7666, fax+7(812) 552-6080.

Поддержка исследований. Работа выполнена при финансовой поддержке РФФИ, проект №11-07-90412-Укр_ф_а.

Рекомендовано лабораторией технологий и систем программирования, зав. лаб. Шкиртиль В.И., к.т.н.

Статья поступила в редакцию 21.02.2013.

РЕФЕРАТ

Ануреев И.С., Баранов С.Н., Белоглазов Д.М., Бодин Е.В., Дробинцев П.Д., Колчин А.В., Котляров В.П., Летичевский А.А., Летичевский А.А. мл., Непомнящий В.А., Никифоров И.В., Потиенко С.В., Прийма Л.В., Тютин Б.В. **Средства поддержки интегрированной технологии для анализа и верификации спецификаций телекоммуникационных приложений.**

В работе описываются разработанные авторами прототипы инструментальных средств и комплексный подход на их основе, при котором методы и средства анализа и верификации обеспечены для представителей всех четырех основных классов языков, на которых обычно описываются телекоммуникационные приложения: языки выполняемых спецификаций общего назначения (SDL), языки для описания и анализа укрупненных образцов поведения и выявления зависимостей между ними в сложных системах (UCM), специализированные языки, ориентированные на верификацию спецификаций телекоммуникационных систем (язык интерпретированных MSC диаграмм, язык взаимодействующих конечных автоматов, язык Dynamic-REAL) и индустриальные императивные языки (C/C++).

Система автоматического преобразования UCM-спецификаций в модель базовых протоколов применяется для последующей их автоматизированной верификации; опробована методика получения с помощью этих средств непротиворечивых и полных поведенческих моделей на ряде представительных телекоммуникационных приложений, включающих как известные телекоммуникационные протоколы, так и конкретные большие программные проекты.

Новая версия программного комплекса SRDSV включает транслятор из языка SDL в язык dREAL, систему автоматического моделирования dREAL-спецификаций и две системы верификации dREAL-спецификаций методом проверки моделей. Новая версия программной системы STSV, которая включает транслятор из SDL в раскрашенные сети Петри и верификатор этих сетевых моделей, базирующийся на методе проверки моделей, использует известную систему CPN Tools для симуляции и анализа РСП.

Верификация спецификаций дополняется автоматизированным построением тестовых наборов, обеспечивающих заданную степень покрытия исходных поведенческих требований с оптимизацией по заданным критериям производительности. Тестовая оболочка позволяет одновременно с прогоном тестов проводить автоматизированный анализ результатов тестирования.

SUMMARY

Anureev I.S., Baranov S.N., Beloglazov D.M., Bodin E.V., Drobintsev P.D., Kolchin A.V., Kotlyarov V.P., Letichevsky A.A., Letichevsky A.A. jr., Nepomniaschy V.A., Nikiforov I.V., Potiyenko S.V., Priyma L.V., Tyutin B.V.
Tools of Integrated Technology for Analysis and Verification of Telecom Application Specs.

The paper describes an integrated approach and respective prototypes of tools which provide methods and means for analysis and verification for representatives of all four major classes of languages used to describe telecom applications: languages of general purpose executable specifications (SDL), languages for high-level descriptions of complex systems behavior patterns and their interconnections (UCM), special purpose language for verification of telecom applications (interpreting MSC, communicating finite state machines, dREAL), and industrial imperative languages (C/C++).

A system of automated transformation of UCM-specifications into a model of basic protocols is used for subsequent verification of these specs by a tool for verification of requirement specificiations based on the theory of insertion programming. The described mehod for obtaining consistent and complete specifications with these tools was tried on number of representative telelcom applications which include known telecom protocols and large industrial projects.

A new version of the program complex SRDSV includes a compiler from SDL into dREAL, a system of automated modeling of dREAL-specifications and two systems for verificaiton of dREAL specifications with model checking. A new version of the program STSV which includes a compiler from SDL into colored Petri nets and a verifier of these network models, also based on model checking, uses a known system CPN Tools for simulations and analysis of colored Petri nets.

Verification of specifications is complemented with automated construction of test suites, which ensure the specified test coverage rate of source behavioral specifications and are optimal with respect to specified performance criteria. Tests are run in a specialized automated test-run environment either on system models, or on actual system implementations inserted in respective program shells which provide communication of the system under test with the test environment. The test shell allows for automated analysis of the test-run results along with the test-runs as well.