Анализ покрытия UCM-модели тестовыми сценариями

П. Д. Дробинцев,
Санкт-Петербургский государственный политехнический университет
drob@ics2.ecd.spbstu.ru

В. П. Котляров
Санкт-Петербургский государственный политехнический университет
vpk@spbstu.ru

И. В. Никифоров
Санкт-Петербургский государственный политехнический университет
i.nikiforov@ics2.ecd.spbstu.ru

***Аннотация* — Рассматриваются подходы анализа покрытия UCM-моделей тестовыми сценариями, сгенерированными по интегральным критериям покрытия. Дается обзор существующих критериев автоматической генерации тестовых сценариев по высокоуровневым UCM-спецификациям. Предлагается два подхода анализа покрытия UCM-модели: автоматический, который предоставляет информацию о покрытых и непокрытых элементах, ветвях и путях в сводной форме, и визуальный, который позволяет пользователю наглядно удостовериться в покрытии UCM-модели. Описанные подходы и концепции реализованы в инструменте анализа, который значительно сокращает время формирования покрывающего множества тестов. Дается направление будущих работ по анализу покрытия сигналов UCM-модели.**

Ключевые слова — критерии генерации, тестовые сценарии, UCM, спецификации, анализ.

I. Введение

В последнее время изменяется традиционная методология разработки программного обеспечения (ПО), в которую включаются этапы детального анализа исходных требований и высокоуровневого дизайна на ранних стадиях разработки. Такое направление развития связано с повышением требований к качеству ПО, при условии удовлетворения исходным спецификациям. Для сокращения времени проверки корректности требований применяют различные технологии автоматизации верификации и тестирования, которые основаны на формальном представлении исходных спецификаций на языках, использующихся как входные для доказательных инструментальных средств.

Одним из наиболее перспективных языков описания требований является язык UCM [1]. Его отличительные особенности простота графической нотации и ориентированность на создание поведенческих сценариев систем. Язык может быть использован для описания требований, как сложных программных продуктов, так и их модулей. При этом язык UCM используется на ранней стадии дизайна перед использованием таких языков, как SDL, MSC, UML. На сегодняшний день одной из наиболее мощных технологий автоматизации верификации и тестирования, которые используют язык UCM, как исходный язык описания спецификаций на систему, является технология VRS/TAT [2].

Отличительными особенностями технологии VRS/TAT является возможность автоматизировать верификацию формальной модели, и по проверенной модели автоматически сгенерировать множество тестовых сценариев. Генерация тестовых сценариев осуществляется по определенному критерию покрытия формальной модели. С этим связана проблема оценки и анализа покрытия исходной формальной модели тестовыми сценариями.

Целью настоящей работы является разработка в рамках технологии VRS/TAT автоматических подходов анализа покрытия формальной UCM-модели тестовыми сценариями для сокращения времени формирования конечного множества покрывающих тестов.

II. Анализ технологий повышения качества программного продукта на оснвое UCM-спецификаций

В настоящее время существуют технологии проектирования формальных систем, использующих стандартизованный язык UCM [1]. К таким технологиям относятся технологии: SPEC-VALUe [3], TR_TROOP [4], USHLTD [5], UCM CPN [6] и технология VRS/TAT [2].

Сравнение технологий проектирования UCM-спецификаций проведено по следующим критериям:

1) соответствие исходного языка спецификаций, т. е. языка спецификаций, используемого в технологии для перехода от текстового неформального описания требований к высокоуровневому дизайну системы, задачам тестирования;

2) возможность использования целевого языка формальной модели, т. е. формального языка, для применения методов доказательства корректности и средств генерации исполнимого кода и тестовых сценариев;

3) применение в сравниваемых технологиях итеративного процесса разработки. Следует отметить, что процесс разработки может быть линейным, т. е. содержать строгую последовательность действий, без внесения

изменений в исходную модель до момента тестирования сгенерированного по модели кода. А может быть итеративным, характеризуемым возможностью корректировки на этапе создания формальной модели, когда анализ, выявление ошибок и исправление исходных спецификаций приводят к повторению процесса доказательства корректности и перегенерации модели;

4) уровень автоматизации работ при проектировании. Технология может быть ручной, полуавтоматической, либо автоматизированной и избавленной от ручной работы по анализу и исправлению найденных ошибок;

5) возможность верификации, характеризуемая наличием поддержки верификации моделей в рассматриваемой технологии;

6) возможность генерации исполнимого кода: поддерживает ли технология генерацию исполнимого кода системы на основе формального описания модели;

7) наличие в технологии опции генерации тестовых сценариев, выражающейся в возможности поддержки технологией генерации тестовых сценариев, используемых для тестирования как модели, так и конечной системы;

8) возможность автоматического исполнения тестовых сценариев.

9) степень поддержки технологией автоматического анализа результатов верификации и тестовой генерации, т. е имеет ли технология автоматизированные средства анализа результатов тестирования и верификации и обратного отображения результатов анализа на исходную модель.

10) наличие в технологии автоматизированного механизма отладки процесса генерации тестовых сценариев.

Сведения по рассмотренным в разделе технологиям по предложенным критериям представлены в таблице I.

Таблица I. . АНАЛИЗ ТЕХНОЛОГИЙ РАБОТЫ С UCM-СПЕЦИФИКАЦИЯМИ

Название Технологии	Критерии сравнения и их реализация в технологиях									
	Исходный язык	Целевой язык	Итеративный процесс	Автоматизированный подход	Верификация	Генерация исполнимого кода	Генерация тестовых сценариев	Исполнение тестовых сценариев	Анализ результатов	Механизм отладки
	1	2	3	4	5	6	7	8	9	10
SPEC-VALUe	UCM	LOTOS	+	-/+	+	-	+	+	-	-/+
RT-TROOP	UCM	MSC/SDL	+	-/+	+	+	-	-	-	-/+
UCM/SDL HLTD	UCM	SDL	+	-/+	+	+	+	-	-/+	-/+
VSR/TAT	UCM	Базовые протоколы	+	-/+	+	+	+	+	-/+	-/+
UCM CPN	UCM	CPN	+	-/+	+	-	-	-	-/+	-

Условные обозначения: «-» – не поддерживается; «-/+» – поддерживается частично; «+» – полностью поддерживается.

Сравнение **по первому критерию** показало, что на ранних этапах разработки важным является использование высокоуровневой нотации для описания модели спецификаций. Все рассмотренные технологии используют UCM как исходный формальный язык спецификаций.

Анализ **по второму критерию**, проводимый с целью выяснения, какой целевой язык используется в технологии, определяет возможности дальнейшей работы со спецификациями. Язык LOTOS не поддерживает возможность генерации кода системы, но позволяет проводить верификацию исходных спецификаций. Язык SDL позволяет проводить как верификацию, так и генерацию исполнимого кода, но его использование затрудняется сложностью понимания сгенерированной модели системы, а также сложностью отслеживания и отображения сценариев поведения системы на исходные модели. Язык базовых протоколов [7], который представляет собой тройку Хоара, состоящую из предусловия, действия выполнимого системой и постусловия, в которое переходит система, позволят проводить, как верификацию модели, так и генерацию тестовых сценариев в формате MSC[8], которые могут быть использованы не только для тестирования, но и для генерации исполнимого кода [9]. Раскрашенные сети Петри (CPN) позволяют проводить анализ системы, но не позволяют генерировать исполнимый код.

Сравнение технологий **по третьему критерию** показало, что все процессы являются итеративными. Все технологии позволяют после анализа модели, выявить недочеты, внести в нее изменения и за несколько итераций добиться необходимой степени корректности.

Анализ технологий **по четвертому критерию** показывает, что все подходы в технологиях являются полуавтоматическими, т. к. требуют либо ручного анализа результатов верификации (технологии SPEC-VALUe, UCM CPN и UCM/SDL HLTD), либо полной автоматизации преобразования исходного языка в целевой (технологии VRS/TAT).

Данные сравнительной таблицы 1 **по пятому критерию** позволяют сделать вывод, что все технологии поддерживают возможность верификации моделей.

Из таблицы 1 по **шестому критерию** следует, что не все перечисленные технологии позволяют генерировать исполнимый код системы. Использование SDL и системы Telelogic позволяет осуществлять генерацию исполнимого кода системы, использование языка базовых протоколов совместно с преобразователем в MSC, также позволяет перейти к SDL модели. Однако технологии SPEC-VALUe и UCM CPN не поддерживают такой возможности, т. к. они направлены на верификацию и генерацию тестовых сценариев, а не исполнимый код системы.

Седьмому критерию удовлетворяют только три технологии — технологии SPEC-VALUe, VSR/TAT, и UCM/SDL HLTD. Они поддерживают возможность генерации тестовых сценариев. Остальные рассматриваемые технологии направлены только на создание верифицируемых моделей систем.

Еще более сложная ситуация возникает при сравнении технологий **по восьмому критерию**. Этому критерию отвечают только две технологии из рассмотренных — технологии SPEC-VALUe и VRS/TAT. На их основе можно производить автоматическое исполнение тестов. Остальные технологии существенно проигрывают по этому критерию указанным технологиям.

Сравнение технологий **по девятому критерию** показало, что только три из них: UCM/SDL HLTD, UCM CPN и VRS/TAT — позволяют осуществить анализ результатов генерации, но при этом данный анализ не полностью автоматизирован и требует ручной работы. Остальные технологии (UCM/SDL HLTD и UCM CPN) не поддерживают концепцию анализа результатов на уровне исходной модели вообще, либо не являются даже частично автоматизированными.

Анализ принятых в технологиях подходов **по десятому критерию** указывает на то, что все технологии, кроме UCM CPN поддерживают механизм отладки процесса генерации, но во всех случаях он делается вручную и не имеет под собой инструментальной поддержки.

Проведенный сравнительный анализ 5 технологий работы с UCM-спецификациями четко указывает на то, что технология VRS/TAT является наиболее мощной из всех рассмотренных, т. к. позволяет не только проводить верификацию модели, но и автоматизировать генерацию и исполнение тестовых сценариев.

III. Критерии генерации тестовых сценариев

Выделим основные критерии генерации тестовых сценариев, которые используются в технологии VRS/TAT.

A. Критерий покрытия UCM-элементов

При данном критерии покрытие происходит по всем элементам UCM-модели [1]. Регистрируется наличие каждого элемента UCM-модели в тестовом сценарии хотя бы один раз. Этот критерий позволяет проверить все состояния программного продукта, но он является наиболее слабым критерием, т.к. не гарантирует проверку требований.

B. Критерий покрытия ветвей UCM-модели

Этот критерий позволят сгенерировать множество тестовых сценариев, которое в совокупности покрывает все ветви UCM-модели. Под ветвью UCM-модели понимается линейный участок поведения, на границах которого присутствуют элементы начала или конца сценария (StartPoint, EndPoint), элементы альтернативного поведения (OrFork, OrJoin, FailurePoint), элементы многопоточного поведения (AndFork, AndJoin), и между границами присутствуют элементы выражающие действия системы (Responsibility).

Генерация тестовых сценариев, удовлетворяющих этому критерию, может быть осуществлена автоматически по исходной UCM-модели без дополнительной информации от пользователя.

C. Критерий покрытия путей UCM-модели

Это критерий прохождения всех возможных вариантов поведения модели. Критерий путей в общем случае не может быть выполнен из-за наличия бесконечного числа вариантов поведения системы [10]. В технологии VRS/TAT этот критерий используется с ограничением по числу итераций цикла, ограничений на интерливинг, а также выбора только интересующих поведений пользователя. Таким образом, для выполнения данного критерия от пользователя требуется формирование дополнительной информации о путях, которые должны присутствовать в тестовых сценариях.

D. Критерий покрытия по GDL-формулам

Это пользовательский критерий покрытия, задаваемый на языке GDL (Guides Definition Language). GDL [11] - язык пользовательских функций, созданный для направленной генерации тестовых сценариев по дереву поведения системы.

Рассмотрим язык GDL подробнее. Он построен на основе определения пользователем формул четырех различных типов: "M", "R", "S" и "T".

- Формулы "M" позволяют определить UCM-подмодель путем указания множества элементов UCM-диаграммы. Таким образом, формула M описывает те элементы, которые будут использованы при обходе UCM-диаграммы.

- Формулы альтернативного выбора ("S") позволяют задать подмножество элементов указанием одного элемента.
- Формулы пользовательских последовательностей ("R") определяют последовательности UCM-элементов, которые должны быть представлены в генерируемом сценарии.
- Формулы "T" определяют совокупность правил описанных в формулах типа “M”, “S”, “R”, которые будет использованы при генерации конкретного тестового набора.

GDL-формулы позволяют в краткой форме описать набор интересующих пользователя поведений системы и произвести генерацию тестовых сценариев, удовлетворяющих данным поведениям.

Процесс генерации тестовых сценариев осуществляется автоматически, независимо от выбранного критерия генерации, но при этом остается необходимость провести анализ получившегося тестового набора для выявления непокрытых частей UCM-модели.

IV. Анализ покрытия UCM-модели тестовыми сценариями

В работе предлагается два способа проведения анализа покрытия UCM-модели тестовыми сценариями. Первый способ заключается в автоматическом обходе тестовых сценариев, сборе информации о покрытых UCM-элементах, UCM-ветвях и UCM-путях исходной формальной модели и представлении непокрытых частей модели в сводной форме. Второй подход заключается в визуальном отображении тестовых сценариев на UCM-модели для того, чтобы пользователь мог наглядно удостовериться в покрытии UCM-модели.

A. Автоматический анализ

Для проведения автоматического анализа покрытия UCM-модели был разработан алгоритм (рис.1), входом которого является множество тестовых сценариев и исходная формальная UCM-модель.

Первым этапом (1) алгоритма является обход UCM-модели и формирование общего множества UCM-элементов и UCM-ветвей. Назовем эти множества U и B соответственно.

Вторым этапом (2) алгоритма является обход всех тестовых сценариев и формирование множества UCM-элементов и UCM-ветвей, содержащихся в тестовых сценариях. Назовем эти множества $\mathbf{T_{эл}}$ и $\mathbf{T_{вт}}$ соответственно.

Эти два множества характеризуют покрытие UCM-модели по критерию элементов и ветвей.

Формально эти множества можно записать как:

$$T_{эл} = \sum_{i=0}^{N}\sum_{j=0}^{M}(E_{ij} \in U),$$

$$T_{вт} = \sum_{i=0}^{N}\sum_{j=0}^{M}(B_{ij} \in B),$$

где N-количество тестовых сценариев, M-количество UCM-элементов тестового сценария, i и j - порядковые номера тестового сценария и UCM-элемента соответственно, U-множество всех UCM-элементов проекта, B-множество всех UCM-ветвей проекта, сформированные на этапе (1).

Рис. 1. Алгоритм формирования сводной таблицы о непокрытых элементах и ветвях UCM-модели.

Третий шаг (3) алгоритма заключается в поиске разницы между множеством всех UCM-элементов проекта (U) и множеством покрытых UCM-элементов ($T_{эл}$) тестовых сценариев, а также между множеством всех UCM-ветвей (B) и множеством всех покрытых тестовыми сценариями UCM-ветвей ($T_{вт}$). Такая разница множеств и показывает непокрытые элементы проекта. Назовем эти два множества $S_{эл}$ и $S_{вт}$ соответственно.

Формально эти разницы можно записать следующим образом:

$$S_{эл} = U - T_{эл}\,, \qquad S_{вт} = B - T_{вт}.$$

Последний четвертый этап (4) алгоритма заключается в формировании сводной таблицы по покрытым и непокрытым элементам и ветвям модели.

Представленный формальный алгоритм вычисления непокрытых элементов и ветвей позволяет автоматизировать процесс анализа и формирования сводной информации о покрытии. Предложенный алгоритм справедлив для анализа покрытия по критериям UCM-элементов и UCM-ветвей.

Рассмотрим правила анализа покрытия по критерию путей и критерию GDL-формул. Как уже говорилось, эти два критерия в общем случае недостижимы, поэтому от пользователя требуется дополнительная информация о

задании интересующих его поведений. Данные поведения задаются в виде последовательности ключевых UCM-элементов, наличие которых в тестовом сценарии обеспечивает покрытие данной последовательности. Такие последовательности называются гидами [12].

Правило: Если все заданные пользователем последовательности обеспечены тестовыми сценариями, то считается что выбранный критерий генерации удовлетворен, иначе - не удовлетворен.

Формально это правило можно записать в следующем виде:

$$\forall G_i \in G \; \exists T_j \in T \Rightarrow K,$$

где G-множество гидов, G_i - i-тый гид, T-множество тестовых сценариев, T_j- j-тый тестовый сценарий, K-выбранный критерий генерации.

Автоматизация анализа по рассматриваемым критериям возможна за счет использования алгоритма поиска вхождения последовательности ключевых UCM-элементов гида в последовательность UCM-элементов трассы.

Отношение вхождения одной последовательности событий в другую было дано в работе [13] и задается выражением:

$$U_m \subset U_n \Leftrightarrow (m \le n) \wedge (\forall i : (1 \le i \le m)$$
$$\exists j_i : ((j_0 = 0) \wedge (j_i > j_{i-1})) \wedge (U_m^i = U_n^{j_i})),$$

где U_m — события критериальной цепочки; U_n — события трассы.

Технология VRS/TAT поддерживает четыре критерия генерации тестовых сценариев, и позволяет автоматически проверить их выполнение на конечном тестовом наборе.

B. Визуальный анализ

Несмотря на наличие автоматизированного подхода вычисления непокрытых элементов и ветвей UCM-модели необходимо предоставить пользователю возможность визуального контроля покрытия непосредственно на UCM-диаграммах. Каждому тестовому сценарию можно задать уникальный цвет, которым он подсвечен на диаграммах исходной модели. Таким образом, можно подсветить на UCM-диаграмме весь тестовый набор для оценки его покрытия.

На рис. 2 представлена UCM-диаграмма с подсвеченным сценарием (он изображен пунктиром), который проходит по двум уровням проекта, организованным через структурный элемент Stub [1].

Возможность визуального изображения сценариев на разных уровнях абстракций системы делает тестовый набор наглядным, что актуально для проектов любой сложности, а также способствует ручному анализу и пониманию UCM-модели.

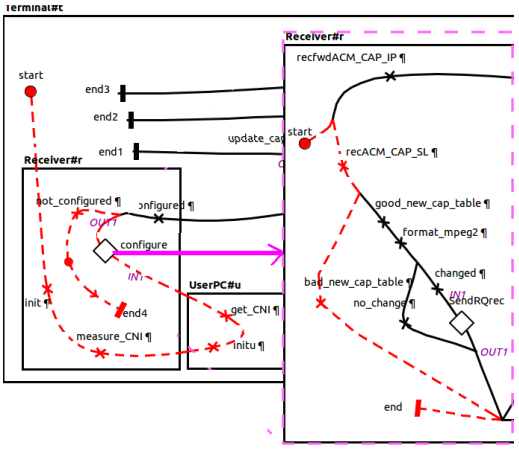

Рис. 2. Визуализация тестового сценария на UCM-диаграмме

V. ПРОВРКА ПОКРЫТИЯ ТРЕБОВАНИЙ ТЕСТОВЫМИ СЦЕНАРИЯМИ

В III и IV рассмотрены критерии генерации тестовых сценариев и способы автоматического и визуального анализа покрытия. Но тестирование, в первую очередь, направлено на проверку удовлетворения разрабатываемой системы исходным спецификациям. Поэтому важной особенностью разработки тестов является возможность создания тестовых сценариев, обеспечивающих 100% покрытие требований. Для этого используется матрица отслеживания [14], представленная на рис.3.

1	2	3	4	5
Идентификатор требования	**Исходный текст требования**	**Сценарий проверки требования**	**Идентификатор сценария**	**Критериальные цепочки**
TRS_18081-2210[7]	Когда установлена опция "Не беспокоить", пользовательский статус отправляется на сервер с сообщением "Не беспокоить"	scen#064: 1. Установить опцию "Do not Disturb" в меню KPTT_MyStatus и нажать LSK "Select" или HK "Center Select" 2. Проверить, что KPTT_Confirmation_Notice отобразилось в меню	scen#064	:TRS_18081_2209 KPTT_MyStatus06 KPTT_Confirmation_N
TRS_18081-2209[6]	Если установлена опция "Не беспокоить", то у пользователя не должно быть возможности организовать PTT	scen#128: 1. Проверить состояние в меню DnD 2. Перейти на KPTT_Contacts в меню, выбрать контакт 3. Нажать на конпку PTT	scen#128	:TRS_18081_2212 KPTT_Contacts39

Рис. 3. Пример матрицы отслеживания

Матрица отслеживания содержит в себе как минимум следующую информацию:

— идентификатор требования (столбец 1);

— неформализованное описание требования (столбец 2);

— сценарий проверки требования (столбец 3);

— идентификатор сценария проверки требования (столбец 4);

— критериальную цепочку [15] событий (столбец 5).

По неформальному описанию требований выделяются сценарии проверки, согласуемые с заказчиком. Сценарии проверки представляют собой последовательность действий пользователя и откликов системы на входные

воздействия, которые необходимо осуществить и наблюдать для доказательства корректности требования.

Затем по сценариям проверки осуществляется разработка критериальных цепочек, которые являются последовательностью основных событий, наблюдаемых в поведении системы при проверке некоторого конкретного требования. Критериальные цепочки не только формализуют требование и сценарий его проверки в последовательности событий, но и являются критерием его проверки.

После того, как в матрице отслеживания появились критериальные цепочки в виде последовательности UCM-элементов, можно переходить к процессу генерации тестовых сценариев по частичному критерию путей. Такой переход становится возможным, т.к. критериальные цепочки задают множество поведений, которые пользователь должен проверить в системе.

Таким образом, можно наблюдать связь между частичным критерием путей и матрицей отслеживания, что в свою очередь обеспечивает 100% проверку требований и позволяет автоматически провести генерацию покрывающего тестового набора.

VI. Инструментальная реализация анализа покрытия

Для решения задач анализа покрытия реализован набор программных средств, интегрированных в инструмент автоматического анализа UCM GA. Инструмент внедрен в автоматическую цепочку тестирования VRS/TAT с целью получения набора тестовых сценариев, покрывающих функциональные требования на проектируемую систему.

Анализатор UCM GA реализует подходы проверки покрытия поведенческой UCM-модели тестовыми сценариями. Он позволяет не только автоматически, но и визуально оценить покрытие поведенческой модели на уровне UCM-диаграмм. Использование анализатора UCM GA позволяет сократить время формирования покрывающего множества тестовых сценариев в 3 раза [16].

VII. Направление будущих работ

При тестировании телекоммуникационных систем, основополагающим способом обмена информацией являются транзакции и сложные сигналы. На сегодняшний день в анализаторе покрытия UCM GA отсутствует возможность проверить наличие всех сложных сигналов разрабатываемой системы и комбинаций их параметров в тестовом наборе, поэтому в направлении будущих работ ставится задача создать метод анализа покрытия сложных сигналов UCM-модели, реализовать его в инструменте автоматического анализа и визуализировать точки обмена сообщениями на уровне UCM-модели.

Список литературы

[1] ITU-T Recommendation Z.151 . User requirements notation (URN), 10/2012

[2] Веселов А.О., Иванов А.С., Тютин Б.В., Котляров В.П. Автоматизация тестирования телекоммуникационных приложений // Научно-технические ведомости СПбГПУ. № 3. СПб.: Изд-во Политехнического ун-та, 2009. С. 208-212..

[3] Amyot D., Specification and Validation of Telecommunication System with Use Case Maps and LOTOS, Ph.D. Thesis, Ottawa, Ontario, Canada, University of Ottawa, September 2001, 128p.K. Elissa, “Title of paper if known,” unpublished.

[4] Bordeleau, F., A systematic and traceable progression from scenario models to communicating hierarchical state machines, Thesis (Ph.D.), Carleton University, 2000.Y. Yorozu, M. Hirano, K. Oka, and Y. Tagawa, “Electron spectroscopy studies on magneto-optical media and plastic substrate interface,” IEEE Transl. J. Magn. Japan, vol. 2, pp. 740-741, August 1987 [Digests 9th Annual Conf. Magnetics Japan, p. 301, 1982].

[5] Sales I., A Bridging Methodology for Internet Protocols Standards Development, Ph.D. Master of Computer Science Thesis, Ottawa, Ontario, Canada, School of Information Technology and Engineering – S.I.T.E., September 30 2001, p.119.

[6] Визовитин Н.В., Непомнящий В.А., Алгоритмы трансляции UCM-спецификаций в раскрашенные сети Петри // Препринтиг 168. – Институт Систем Информатики им. А.П. Ершова. – Новосибирск. – 2012. [Электронный ресурс]. Режим доступа: // www.iis.nsk.su/files/preprints/168.pdf

[7] Летичевский А.А. Верификация и тестирование интерактивных систем, специфицированных базовыми протоколами. Диссерт. на соискание уч. ст. канд. физ.-мат. наук. Киев, 2005. 138 с.

[8] ITU Recommendation Z.120. Message Sequence Charts (MSC), 11/99.

[9] Letichevsky A., Kapitonova J., Letichevsky Jr., A., Volkov V., Baranov S., Weigert T. Basic protocols, message sequence charts, and the verification of requirements specifications, Computer Networks: The International Journal of Computer and Telecommunications Networking, v.49 n.5, 5 Dec 2005. P. 661-675.

[10] Beizer B., Software Testing Techniques, 2nd Edition, New York, 1990, P.550

[11] Дробинцев П.Д., Никифоров И.В., Котляров В.П. Методика проектирования тестов сложных программных комплексов на основе структурированных UCM моделей // Научно-технические ведомости СПбГПУ. № 3(174). СПб.: Изд-во Политехнического ун-та, -2013. -С. 99-105

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

[13] Баранов С.Н., Котляров В.П. Формальная модель требований, используемая в процессе генерации кода приложения и кода тестов // Моделирование и анализ информационных систем, 2011, том 18, №4, С.118–130.

[14] Drobintchev P., Kotlyarov V., Nikiforov I., “Technology Aspects of State Explosion Problem Resolving for Industrial Software Design”// Proceedings of the 7th Spring/Summer Young Researchers’ Colloquium on Software Engineering, Kazan, May 30-31, 2013, pp. 46-51

[15] Drobintsev P.D., Nikiforov I.V., Kotlyarov V.P., Semantics adjustment of UCM Real Time Constructions for Implementation in Translator of UCM to Basic Protocols // Humanities and Science University Journal. № 5 (2013): научный журнал / Санкт-Петербургский университетский консорциум. – СПб., 2013. - С. 207-215.

[16] Никифоров И.В. Методы автоматизации построения поведенческой модели программного продукта на основе UCM-спецификаций Диссерт. на соискание уч. ст. канд. тех. наук. СПб.: СПбГПУ, 2013. 205 с.