УДК 004.896

В.А. Лазарев

МЕТОДИКА РАЗРАБОТКИ СРЕДСТВ ИНТЕЛЛЕКТУАЛЬНОЙ ПОДДЕРЖКИ АВТОМАТИЗИРОВАННОЙ СИСТЕМЫ ТЕСТИРОВАНИЯ ПРОГРАММНЫХ КОМПЛЕКСОВ

«Интел ЗАО», Нижний Новгород

Представлен результат анализа предметной области систем автоматизированного тестирования и рассматривается методика интеллектуальной поддержки подобных систем с использованием продукционной модели представления знаний.

Ключевые слова: Интеллектуальная поддержка, продукционная модель, автоматическое тестирование

По мере развития индустрии программного обеспечения (ПО) развиваются и средства обеспечения заданного качества выпускаемых продуктов. Одним из основных подходов к достижению заданного качества продукта стала автоматизация процесса тестирования программного обеспечения. Данный сегмент рынка автоматизированных систем (АС) активно развивается. Однако, к сожалению, не существует единого формализованного подхода к комплексу вопросов построения и поддержки данного класса систем.

Как показала практика, применение автоматизированного тестирования несет ряд сложностей, проявляющихся по мере их развития:

- увеличение количества тестов приводит к увеличению машинного времени требующемуся для получения тестовых результатов, а также к увеличению времени анализа полученных результатов;
- гетерогенность тестовых платформ, как программная, так и аппаратная, требует различных подходов и компетенций для анализа тестовых результатов;
- основной фокус при анализе тестовых результатов делается внутрь теста. Проблемы, привносимые окружением, чаще всего остаются «за скобками»;
- большой объем рутинных действий приводит к демотивации персонала, что негативно сказывается на качестве анализа тестовых результатов.

В данной работе ставится задача автоматизации процесса поддержки автоматизированного тестирования программных комплексов. Для этого приводятся результаты анализа предметной области функционирования рассматриваемых АС. Предлагается использовать интеллектуальную систему (ИС), в основе которой лежит продукционная модель представления знаний в качестве инструментария поддержки системы автоматизации тестирования программных комплексов (САТПК). Представлена методика формализации знаний о предметной области и построения инструментариев интеллектуальной поддержки.

Теоретический анализ

Анализ предметной области их функционирования позволил сформулировать ряд типовых требований к САТПК.

1. Поддержка в режиме реального времени любых изменений в тестируемом ПО. Реализация этих идей позволяет выявлять проблемы в программном продукте на ранних стадиях разработки ПО. По мере роста размеров и сложности программных продуктов количество тестов и время, требуемое для их исполнения, растет пропорционально. В результате для получения результатов за ограниченное время требуется увеличивать парк тестовых машин и распараллеливать процесс тестирования.

2. Предоставление достоверных данных. Неверные результаты тестирования приводят к дополнительным расходам (либо на анализ ошибки в тесте, либо на исправление ошиб-

ки на более поздних этапах жизненного цикла программного продукта, что всегда «дороже»). Кроме того, аппаратные или программные сбои на конкретных машинах могу заблокировать процесс тестирования. Для решения подобных проблем увеличивают количество тестовых машин в каждой платформе. В результате подозрительный тест можно выполнить на нескольких однотипных тестовых машинах, что зачастую помогает локализовать проблему.

3. Гетерогенность тестовых платформ. Так как многие программные продукты поддерживают несколько платформ (как минимум несколько версии операционной системы), требуется проводить тестирование на парке тестовых машин.

4. Непрерывная работа САТПК. Для сокращения издержек требуется максимально загрузить имеющиеся мощности, в том числе в нерабочее время.

5. Эта задача решается путем автоматизации запуска, распределения тестовых задач, а также автоматизации сбора тестовых результатов.

6. Отслеживаемость процесса тестирования. При автоматическом исполнении тестов необходимо иметь возможность анализа тестового результата по завершении тестов. Для этого все данные о настройке тестового окружения и самом процессе выполнения теста должны сохраняться, включая информацию в стандартных потоках ввода-вывода, статус промежуточных операций.

7. Воспроизводимость результатов. Для анализа тестового падения полезно иметь возможность перезапустить упавший тест с той же версией тестов и продукта, а также на той же программно-аппаратной платформе.

8. Автоматизация процесса построения тестов и тестируемого ПО.

9. Поддержка в режиме реального времени процесса автоматического тестирования. Статистически выявлено, что эффективная поддержка работы САПТК способна без дополнительных затрат поднять эффективность работ бизнес-единицы, которая разрабатывает и тестирует ПО.

Большинство из сформулированных требований воспроизводится в каждой из известных авторам САТПК. Это позволило формализовать типовую архитектуру данного класса АС, которая представлена на (рис. 1).

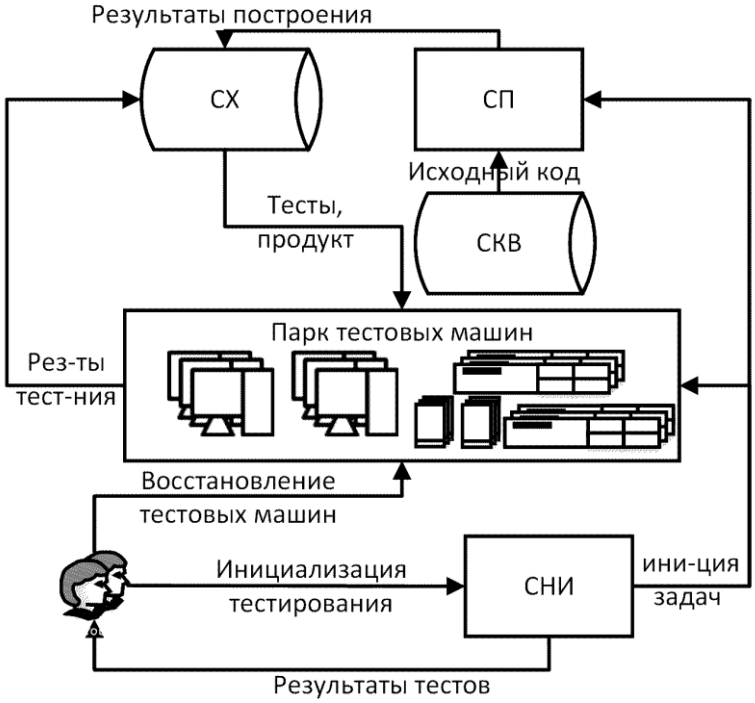

Рис. 1. Типовая архитектура САТПК

На рисунке представлена общая архитектура системы автоматического тестирования,

состоящая изпарка тестовых машин, системы распределения задач, в качестве которой используется система непрерывной интеграции (СНИ), системы построения (СП), системы контроля версий исходного кода (СКВ), сетевого хранилища (СХ), персонала, который обеспечивает работу системы.

В данной работе предлагается внедрить систему интеллектуальной поддержки процесса автоматического тестирования. В результате произойдет переход к системе, указанной на рис. 2, в которой интеллектуальный модуль (СИП) автоматизирует ряд рутинных операций, которые ранее требовали участия персонала.

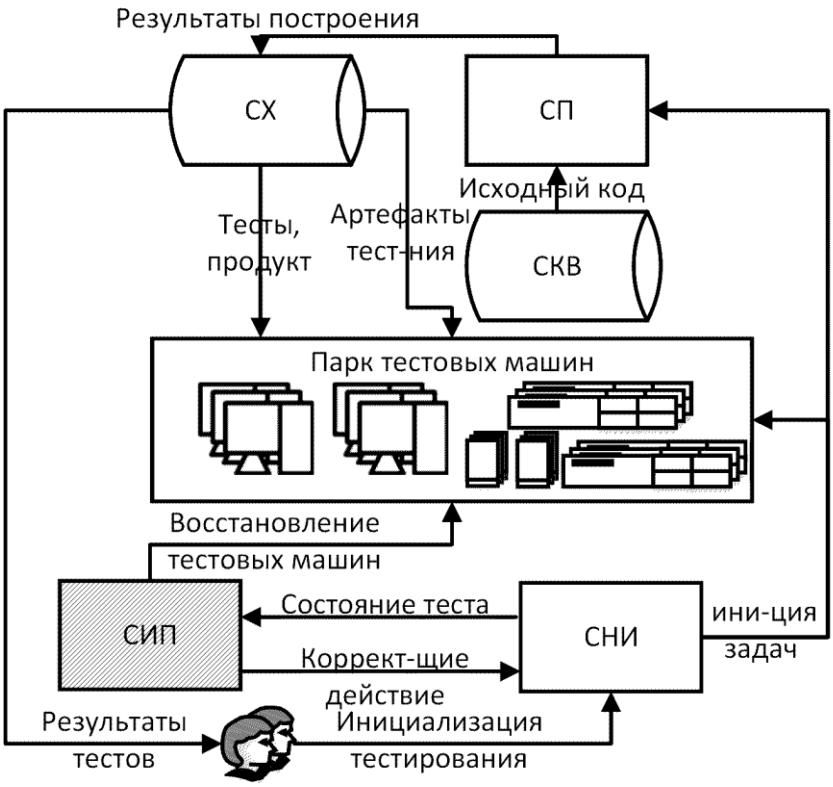

Рис. 2. Модифицированная архитектура САТПК

Методика

Для перехода к данной архитектуре требуется формализовать процедуру представления данных о предметной области работы СИП и ее представление в машинно-ориентированном виде. В качестве ядра СИП выступает продукционная экспертная система, построенная с помощью оболочки CLIPS [1].

Выбор продукционной модели обусловлен следующими свойствами [2]:

- естественность для человека. Подобные системы идеально моделируют логическое мышление без учета эмоций;
- ориентация на модифицируемость и расширяемость. Данное свойство обусловлено:
  - аддитивным характером правил. Т.е. если к множеству правил добавить новое подмножество, изначальное множество не меняется;
  - локальностью изменения. При изменении отдельного правила меняется только одна причинно-следственная связь;
- машинная ориентация правил. Продукционные правила легко реализуются с помощью баз данных и композитной логики.

Кроме того, в пользу выбора продукционной модели говорят следующие свойства предметной области:

- основными знаниями являются информация о причинно-следственных связях между

различными состояниями элементов подсистем (исполняющийся тест, тестовая машина, результат конкретного теста) системы автоматического тестирования и способы приведения их в желаемое состояние. Основной задачей системы поддержки автоматического тестирования является управление работой элементов различных подсистем. Для поддержания их в требуемом состоянии наиболее логично использовать продукционные правила ЕСЛИ…ТО…;

- тестовые инженеры чаще всего используют неформализованные процедуры в формате: ЕСЛИ…ТО… Данные процедуры легко формализуются с помощью продукционных правил.

В результате применение продукционных правил позволит эффективно построить базу знаний поддержки системы автоматического тестирования. В процессе анализа знаний о предметной области формат продукционного правила претерпел ряд трансформаций:

1. В простейшем случае продукционной модели правило имеет следующий вид:

$$(k)\text{ЕСЛИ } A_i \text{ и} \dots \text{и } A_j \text{ ТО } B_k,$$

где k – идентификатор правила;

A – множество допустимых условий;

B – множество допустимых действий.

2. Для упрощения поддержки и минимизации вероятности возникновения формальных и концептуальных конфликтов, свойственных продукционной модели, БЗ разбивается на фрагменты в соответствии с типовыми задачами в предметной области. Правило примет следующий вид:

$$(k)\text{ЕСЛИ } (frag) \text{ и } (A_i \text{ и} \dots \text{и } A_j) \text{ ТО } B_k,$$

где frag – идентификатор фрагмента предметной области, к которой принадлежит данное правило.

3. Для разрешения формальных и концептуальных конфликтов, свойственных продукционным БЗ, а также для оптимизации БЗ вводятся функции мер:

$$(k)\ (M_{k,1} \dots M_{k,n})\ \text{ЕСЛИ }\ (frag) \text{ и } (A_i \text{ и} \dots \text{и } A_j) \text{ ТО } B_k,$$

где M – множество допустимых мер, применяемых к правилам.

4. Для решения проблемы сложности представления всего множества параметров реальных объектов, вводятся дескрипторы, которые указывают на паспорт ситуации:

$$(k)\ (D.M_1 \dots D.M_n)\ \text{ЕСЛИ }\ (frag) \text{ и } (D.A_i \text{ и} \dots \text{и } D.A_j) \text{ ТО } D.B_k,$$

где D –дескриптор ситуации.

Данный дескриптор ссылается на паспорт ситуации, содержащий всю информацию, требуемую для работы правила. Пример паспорта ситуации для правила «проверка сетевой доступности тестового устройства» приведен на рис. 3.

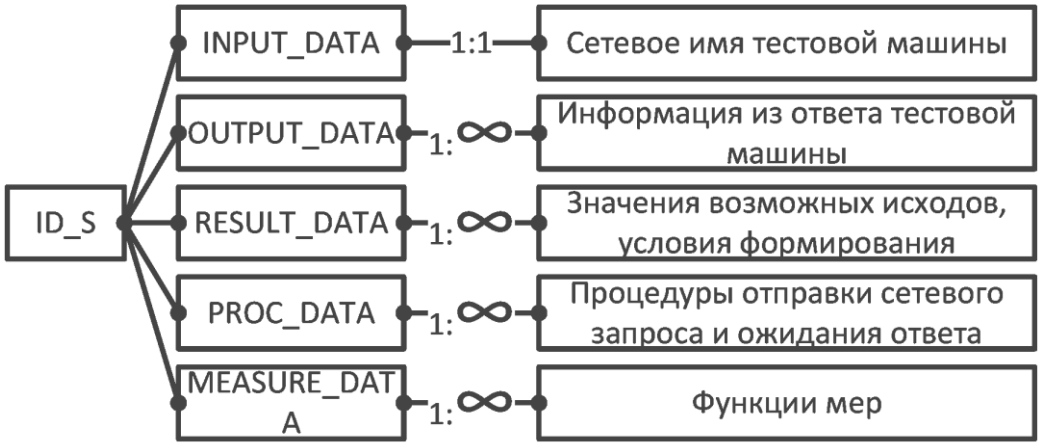

Рис. 3. Пример паспорта ситуации

Предлагается использовать модифицированный формат продукционного правила. Для формализации знаний о предметной области и их представления в заданном формате используется модель «конструктор», предложенная в работе [3]. В рамках такой модели проектирование осуществляется в соответствии со стратегией «сверху - вниз». При этом вводится четывре уровня проектирования (рис. 4).

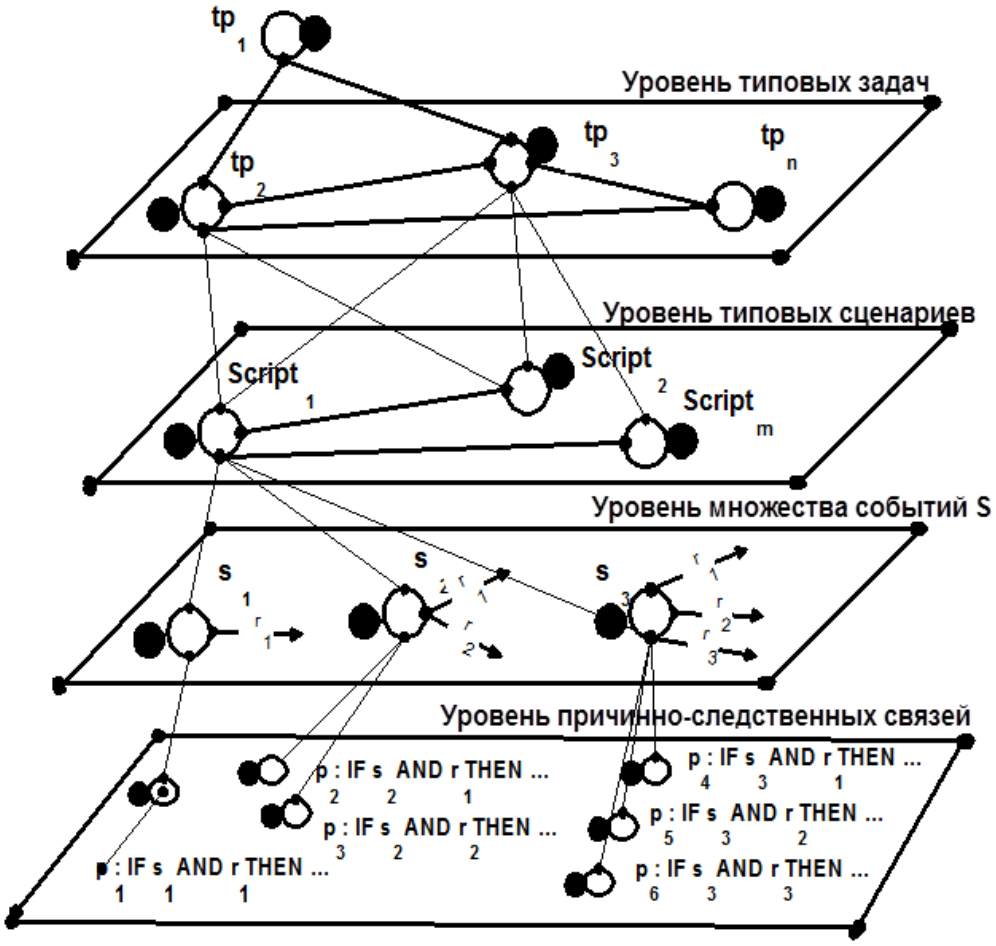

Рис. 4. Модель конструктор

Уровень типовых задач описывает набор обособленных подзадач, требующих решения для решения глобальной задачи. Уровень типовых сценариев описывает множество сценариев, позволяющих решать типовые задачи. Уровень событий описывает объекты управления и их параметры, необходимые работы сценариев. Уровень причинно-следственных связей содержит набор продукционных правил, являющихся основой для вышестоящих уровней. В результате проектирование интеллектуального модуля разбивается на следующие этапы.

В качестве первого шага строится множество типовых задач в предметной области. Совокупность элементов типовых задач образует класс, которому в абстрактной модели соответствует родовой объект «Типовая задача». Далее между элементами класса устанавливаются горизонтальные, вертикальные и причинно-следственные связи. Следующим шагом является построение для каждой типовой задачи множества типовых сценариев ее решения. Для этого производится декомпозиция предметной области на элементы порождающего сценарии множества событий. При этом каждая типовая задача рассматривается отдельно, что позволяет упростить анализ и формализацию. Пересечения на уровне типовых сценариев или типовых событий искусственно убираются за счет дублирования объектов пересечения.

После проектирования сценарной части осуществляется формирование ситуационных портретов. Для этого целесообразно использовать стратегию «снизу - вверх». При этом в ка-

честве первого шага осуществляется проектирование СП для элементов порождающего сценарии множества событий. Затем генерируются ситуационные портреты для типовых сценариев и, наконец, ситуационные портреты для типовых задач.

Для автоматизации формирования ситуационной части предлагается использовать алгебру ситуационных портретов. Этот аппарат позволяет автоматизировать процесс формирования СП для порождающего сценарии множества событий, сценариев, типовых задач.

На заключительном этапе происходит переход от концептуального описания к машинно-ориентированному. При этом сценарии отображаются в базу правил продукции, а состояния в базу фактов.

Экспериментальная часть

Проиллюстрируем применение данного подхода проектированием фрагмента базы знаний (БЗ), описывающей одну из типовых задач системы поддержки автоматического тестирования. На первом шаге проектирования модели СИП САТПК выделим множество типовых задач, которые решаются в рассматриваемой предметной области $TP = \{tp_1, tp_2 ... tp_n\}$.

Рассмотрим типовые задачи, возникающие при поддержке САТПК.

Таблица 1

Типовые задачи

Типовая задача
Управление состоянием парка тестовых и служебных машин
Управление конфигурацией тестов
Управление запуском набора тестов
Управление процессом исполнения тестов
Управление процессом сохранения тестовых результатов
Автоматический анализ тестовых результатов и связь их с системой учета ошибок
Анализ статистики запусков и генерация рекомендаций по оптимизации тестирования
Генерация новых правил и знаний

На втором шаге требуется выделить набор событий, составляющих сценарий решения типовой задачи. В таблице представлены события, из которых формируется обобщенный сценарий анализа состояния тестового устройства.

Таблица 2

События для типовой задачи «контроль состояния тестового устройства»

Обозначение параметра	Описание события в рассматриваемой предметной области	Количество исходов
s_1	Состояние тестового устройства неизвестно	1
s_2	Тесты исполняются на устройстве?	2
s_3	Тестовое устройство отвечает на сетевые запросы?	2
s_4	Тестовые результаты доступны?	2
s_5	Причины падения тестов?	4
s_6	Доступность в СНИ?	2
s_7	Клиент СНИ настроен корректно?	2
s_8	Причина недоступности устройства в СНИ?	5

На следующем шаге осуществляется ранжирование событий, для чего:

- на первом этапе ранжируются исходы в соответствии с вероятностью его наступления на основе статистики предыдущих запусков автоматизированных тестов (табл. 3);

- на втором этапе ранжируются события в соответствии с статистическим критерием, вычисляемым как сумма вероятностей возможных исходов (табл. 4).

Таблица 3

Вероятность возникновения исходов

Обозначение состояния	Описание состояния	Вероятность возникновения
sr_1	Тестовые результаты доступны, целевое состояние.	89,10%
sr_2	Недостаточный уровень привилегий.	0,09%
sr_3	Некорректное окружение.	0,18%
sr_4	Недоступны сетевые диски.	0,04%
sr_5	Проблема СНИ.	0,13%
sr_6	Не отвечает на сетевые запросы.	2,67%
sr_7	Клиент СНИ сконфигурирован, но не исполняется.	0,09%
sr_8	Клиент СНИ не сконфигурирован.	0,03%
sr_9	Отключен пользователем.	0,89%
sr_{10}	Отключен из-за проблем с ресурсами.	0,53%
sr_{11}	Идет процесс зеркалирования.	1,78%
sr_{12}	Идет тест, требующий эксклюзивного доступа к устройству.	4,46%

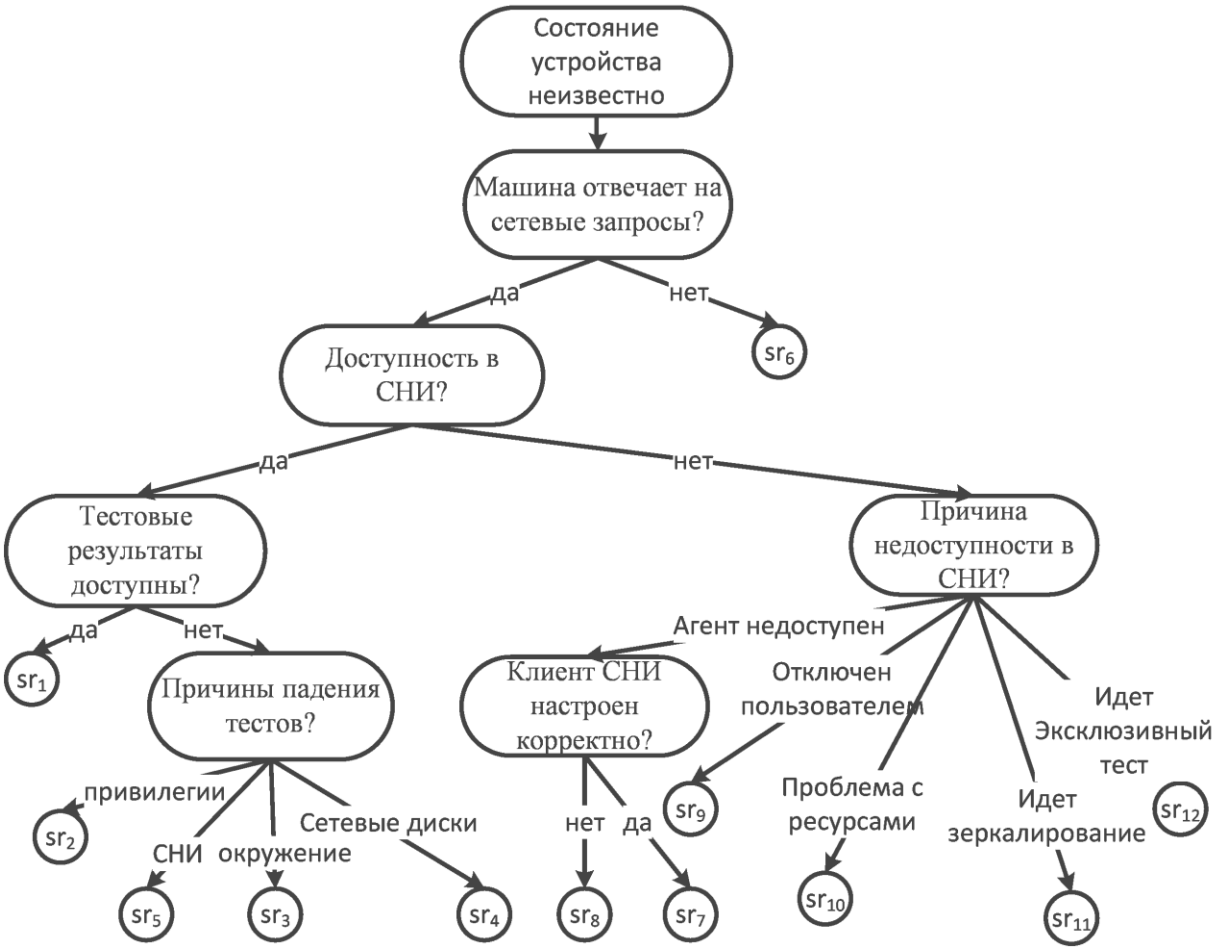

Рис. 5. Сценарий «анализ состояния тестового устройства»

Таблица 4

Вероятность успешного применения события

Обозначение параметра	Описание (параметра объекта) в рассматриваемой предметной области	Вероятность события
s_1	Состояние тестового устройства неизвестно	100%
s_2	Тесты начинают исполнение на устройстве?	89,55%
s_3	Тестовое устройство отвечает на сетевые запросы?	100%
s_4	Тестовые результаты доступны?	89,10%
s_5	Причины недоступности тестовых результатов?	0,44%
s_6	Доступность в СНИ?	100%
s_7	Клиент СНИ настроен корректно?	0,12%
s_8	Причина недоступности устройства в СНИ?	7,63%

Строится дерево сценария в соответствии с приоритетом событий (рис. 5).

На следующем этапе создаем правила в формате системы CLIPS:

```
(defrule check-ci-agent “Клиент СНИ настроен корректно?”
        (pool-monitoring)
        (measure1 first)
    (not (repair ?))
=> measure1 second
        (if ci-agent-is-configured
        then (repair start-ci-agent)
        else (repair configure-ci-agent))
```

Результаты

Предложена модифицированная продукционная модель описания процессов в предметной области функционирования САТПК. Разработано ядро инструментального комплекса поддержки процесса автоматизированного тестирования на всех этапах ЖЦ. Полученные результаты позволили перейти к новой архитектуре, в которой средства интеллектуальной поддержки дополняют инженеров, что позволяет:

- повысить качество анализа тестовых результатов,
- снизить требования к квалификации персонала,
- сократить время затрачиваемое на поддержку автоматического тестирования.

Перспективы дальнейшего развития:

- автоматизация процедуры выделения новых знаний и их представления в машинно-ориентированном виде, а также их верификации [4–6];
- доработка инструментального комплекса – скелетной оболочки поддержки системы АТПК.

Библиографический список

1. A Tool for Building Expert Systems: [электронный ресурс]: URL: http://clipsrules.sourceforge.net/ (дата обращения 21.09.2014)
2. **Лорьер, Жан-Луи.** Системы искусственного интеллекта // Курьер. 1991. С. 379–400.
3. **Мисевич, П.В.** Сценарно-ситуационный подход к проектированию средств интеллектуальной поддержки процесса функционирования автоматических систем // Системы управления и информационные технологии. 2007. №2.1(28). С. 166–171.

4. **Милов, В.Р.** Программный комплекс автоматизированной верификации реализаций протокольных объектов / В.Р. Милов [и др.] // Системы управления и информационные технологии. 2014. №4 (58). С. 35–41.
5. **Баранов, В.Г.** Имитатор передачи сообщений автоматического зависимого наблюдения для управления воздушным движением / В.Г. Баранов [и др.] // Системы управления и информационные технологии. 2014 №3.0 (57). С. 49–53.
6. **Белов, Д.А.** Проблемно-ориентированная автоматизированная система мониторинга движения железнодорожного состава / Д.А. Белов, П.В. Мисевич, В.П. Хранилов // Автоматизация в промышленности. 2009. №2. С. 49–51.

Дата поступления
в редакцию 16.04.2015

V.A. Lazarev

TOOLS DESIGN METHODOLOGY FOR INTELLECTUAL SUPPORT OF AUTOMATED SOFTWARE TESTING SYSTEM

«Intel CJSC », Nizhny Novgorod

Purpose: Growth of test automation systems highlights need in support automation for them.
Design/methodology/approach: The paper introduces new architecture of software testing system containing intellectual support module. It provides methodology for analysis of object domain and developing the intellectual module built on production knowledge base with modified format of rules. «Constructor» model is used as design approach.
Findings: It is possible, to apply quality of tests automation systems by introducing of intellectual support module.
Research limitations/implications: The present study provides a starting-point for furtherresearch in the area of automated knowledge detection, formalization and verification.
Originality/value: Presented approach allows improve quality of tests analysis, reduces time needed for test cycle execution and decreases requirements for test engineer qualification.

Key words: intellectual support, production model, automated testing, “Constructor” model.