УДК 002.56+621.382

Д.Б. БУЙ, В.Г. СКОБЕЛЕВ

Киевский национальный университет имени Тараса Шевченко, г. Киев, Украина
Институт прикладной математики и механики НАН Украины, г. Донецк, Украина

БЕЗОПАСНОСТЬ ПРОГРАММНЫХ СРЕДСТВ: МОДЕЛИ И МЕТОДЫ (ОБЗОР)

Работа представляет собой краткий обзор моделей и методов, предназначенных для обеспечения безопасности программных средств современных компьютерных систем на протяжении всего их жизненного цикла. Рассмотрены подходы к разработке технологий, обеспечивающих безопасность программных средств и возникающие при этом сложности. Охарактеризована общая схема, применяемая для построения системы защиты программных средств. Представлены основные типы систем защиты программных средств и выделены типы методов, используемых при их построении. Охарактеризованы модели и методы, применяемые в процессе верификации программных средств. Указаны некоторые возможные направления исследований, связанных с построением формальных моделей и методов верификации программных средств. Кратко рассмотрены методы обеспечения защиты баз данных.

Ключевые слова: *программные средства, системы защиты программных средств, верификация*

Введение

Известно, что обеспечение надежности процесса сбора, накопления, обработки, передачи и хранения информации является одной из наиболее актуальных проблем для компьютерных систем (КС) с критической областью применения. Требования, предъявляемые к безопасности КС, устанавливаются рядом международных стандартов. На их основе с учетом особенностей конкретного проекта формируется технология обеспечения безопасности КС на протяжении всего их жизненного цикла. Существенной составляющей такой технологии является технология создания программных средств (ПС) (т.е. программного обеспечения (ПО) и баз данных (БД)), обеспечивающая безопасность функционирования ПС, т.е. функциональную, технологическую и эксплуатационную безопасность ПС [1-10]. Эти три аспекта характеризуются следующим образом:

1.Функциональная безопасность ПС состоит в их способности противостоять непреднамеренным дестабилизирующим факторам, основными из которых являются отказы внешней аппаратуры, аппаратуры КС и средств телекоммуникации, искажение исходной информации от внешней среды и потребителей обработанной информации, дефекты и ошибки ПС, недостатки средств обнаружения опасных отказов и оперативного восстановления ПС.

2. Эксплуатационная безопасность ПС состоит в их способности обеспечить безопасность информации в процессе штатной эксплуатации КС.

3. Технологическая безопасность ПС состоит в их способности предотвращать в процессе функционирования КС воздействия, осуществляемые в целях несанкционированного раскрытия, изменения или уничтожения информации.

Отметим, что между этими тремя аспектами имеются глубокие логические связи (например, нарушение технологической безопасности ПС может повлечь за собой нарушение их функциональной или эксплуатационной безопасности).

В настоящее время проблемы обеспечения функциональной и технологической безопасности ПС изучены недостаточно и на их решение направлены основные усилия исследователей, а проблема обеспечения эксплуатационной безопасности ПС достаточно полно проработана и постепенно отходит на второй план [1,11-16].

Сложность разработки технологий, обеспечивающих безопасность функционирования ПС, обусловлена, по крайней мере, следующими двумя причинами. Во-первых, комплексы ПС современных КС являются одними из наиболее сложных систем, созданных человеком. Во-вторых, такие технологии должны охватывать весь жизненный цикл ПС, т.е. системный анализ проекта, проектирование, разработку, тестирование, испытания, поддержку эксплуатации и сопровождение ПС, модификацию и создание новых версий.

Значение технологий, обеспечивающих безопасность функционирования ПС, характеризуется тем, что суммарные затраты на создание, развитие и поддержку ПС превосходят затраты на аппаратное обеспечение КС, а ущерб от использования некачественных ПС приводит к катастрофическим последствиям и астрономическим финансовым затратам. Это обстоятельство стимулировало появление следующих методологий [17-19], обеспечивающих по-

тенциальную возможность реализации указанных технологий разработки ПС.

1. Microsoft Solutions Framework (MSF). Методология основана на принципах работы с продуктами фирмы Microsoft и предназначена для ведения проектов и принятия решений на основе адаптируемой модели коллектива разработчиков, поддерживаемой программной средой разработки Microsoft Visual Studio 2005 Team Edition. Проект разработки ПС реализуется поэтапно с наличием соответствующих контрольных точек (водопадная модель), а сама последовательность этапов может повторяться (спиральная модель).

2. Rational Unified Process (RUP). Проект оформляется в виде размещенной на Web базы знаний, снабженной поисковой системой. Методология обеспечивает распределение задач и ответственности в команде разработчиков ПС, поддерживается средствами автоматизации отдельных этапов разработки ПС и предоставляет лучшие (из накопленного опыта разработки ПС) руководства, шаблоны и рекомендации по использованию инструментальных средств для всех критически важных работ в течение всего жизненного цикла ПС.

3. EXtreme Programming (XP). Методология ориентирована на обеспечение эффективного взаимодействия как внутри команды разработчиков, так и с заказчиком, выработку наиболее простых работающих решений, гибкое адаптивное планирование, предназначенное для постепенного улучшения разрабатываемых ПС за счет коротких итераций, основанных на выполнении очередной части требований заказчика.

Основными причинами, приводящими к неприемлемым результатам в процессе функционирования КС, являются сбои, ошибки программистов и операторов, преднамеренные и непреднамеренные дефекты ПС. Большинство этих причин может быть устранено за счет комплексного тестирования, верификации и валидации ПС в процессе их разработки и реализации.

Тестирование ПС представляет собой управляемое исполнение программ, предназначенное для проверки их корректности, т.е. для обнаружения ошибок и несоответствий поведения требованиям. Выделяют следующие типы тестирования ПС [20]:

1. Модульное тестирование. Выполняется независимо для каждого небольшого модуля. Предназначено для покрытия путей в графе зависимости данных, логических ветвей, граничных значений параметров.

2. Интеграционное тестирование. Выполняется для наращиваемых подсистем ПС, получаемых при последовательном добавлении групп модулей. Предназначено для проверки корректности функционирования системы модулей, образующих тестируемую подсистему ПС.

3. Системное тестирование. Выполняется для всей системы ПС. Предназначено для проверки корректности реализации ее функций, производительности, времени отклика, устойчивости к сбоям, атакам, ошибкам пользователя.

4. Нагрузочное тестирование. Выполняется для всей системы ПС при имитации различных возможных нагрузок на нее. Предназначено для выработки наиболее приемлемых архитектурных решений КС, а также для проведения приемно-сдаточных испытаний в условиях, приближенных к реальным.

Понятия верификации и валидации ПС последовательно уточнялись стандартами IEEE 610.12-1990 [21], IEEE 1012-2004 [22] и ISO/IEC 12207 [23]. В настоящее время они понимаются следующим образом.

Верификация ПС представляет собой проверку соответствия их характеристик заданным требованиям и стандартам в процессе разработки и сопровождения, а также анализ причин возникновения ошибок и последствий, которые вызовет их исправление. Таким образом, тестирование ПС, если из него удалить методы, направленные непосредственно на отладку программ, представляет собой один из видов верификации. Отметим, что верификация ПС выполняется всегда, а методы ее осуществления существенно используют формальные модели [24].

Валидация ПС представляет собой проверку соответствия их характеристик потребностям пользователей и заказчиков с учетом предметной области и ограничений контекста использования ПС. Она выполняется только при необходимости, причем с участием представителей заказчиков, пользователей и экспертов в предметной области. Методы ее осуществления часто используют специфические техники выявления знаний и действительных потребностей участников. Отметим, что валидация является значительно менее формализованной деятельностью чем верификация.

Таким образом, говоря неформально, верификация ПС является доказательством того, что ПС разрабатываются правильно, а валидация ПС является обоснованием того, что разрабатывается правильное ПС.

Преднамеренные дефекты могут быть внесены как при создании ПС, так и при их эксплуатации. При создании ПС они могут быть внесены разработчиками алгоритмов и программ, спецслужбами и инструментальными средствами проектирования, автоматически генерирующими деструктивные программы. При эксплуатации ПС они могут быть внесены деструктивными программами или программистами, имеющими опыт разработки и отладки

программ на уровне ассемблера. В последнем случае внесение дефекта часто осуществляется в соответствии со следующей схемой: дизассемблирование исполняемого программного кода, получение исходного текста, внесение в него деструктивной программы, повторная компиляция, корректировка идентификационных признаков программы (чтобы новая программа была схожа с оригиналом).

Одним из основных источников преднамеренных дефектов ПС являются алгоритмические и программные закладки. Под алгоритмической закладкой понимают преднамеренное завуалированное искажение алгоритма, либо его построение таким образом, что в результате программной реализации будут возникать непредусмотренные ограничения на выполнение требуемых функций или, наоборот, в программе возможно появление функций, не предусмотренных спецификацией. Под программной закладкой понимают преднамеренное завуалированное внесение в программу фрагментов, реализующих непредусмотренный алгоритм, который ограничивает выполнение программой требуемых функций, либо, наоборот, реализует функции, не предусмотренные спецификацией.

Опасность программных закладок состоит в том, что в них может быть заложено управление извне, приводящее при определенных условиях к блокированию работы всей КС. Таким же свойством могут обладать электронные закладки в импортной вычислительной технике. Поэтому использование импортной техники, а также импортных ПС и информационных технологий при создании КС с критической областью применения заведомо несет в себе потенциальную угрозу с непредсказуемыми последствиями.

Отметим, что риск угрозы КС существенно возрастает из-за использования нелицензионных ПС, что обусловлено, по крайней мере, следующими тремя причинами. Во-первых, отсутствует их техническая поддержка со стороны производителя. Во-вторых, в них возможно наличие деструктивных программ. В-третьих, при переделке (взломе) лицензионной версии могут быть удалены фрагменты, предназначенные для защиты ПС, а также фрагменты, обеспечивающие стабильность работы ПС.

1. Общая схема разработки системы защиты ПС

Основными этапами разработки системы защиты ПС являются следующие [25-28]:

1) выделение характеристик, влияющих на безопасность функционирования ПС;

2) выбор принципов обеспечения безопасности функционирования ПС;

3) определение категорий отказов, определяющих безопасность функционирования ПС;

4) выделение уровней безопасности ПС в зависимости от ситуаций, возникающих при отказах;

5) определение типов внешних и внутренних угроз безопасности функционирования ПС;

6) распределение ресурсов, предназначенных для создания системы защиты ПС;

7) выбор и реализация системы защиты ПС.

Рассмотрим эти этапы более подробно.

Характеристики, влияющие на безопасность функционирования ПС, выделяются, исходя из потребностей эффективной реализации КС и основных функций ПС при реальных угрозах. На их основе устанавливается необходимый уровень безопасности функционирования ПС. Выбираются методы и средства его обеспечения. Сложность такого выбора обусловлена тем, что их реализация осуществляется при ограниченных ресурсах.

Особую роль играют характеристики внешней среды. На их основе определяется адекватная исходная информация от объектов внешней среды, необходимая для реализации функций, определенных требованиями к КС, осуществляется выбор средств физической защиты ПС и средств защиты ПС от неадекватных действий персонала. Внешняя среда имеет три изменяющиеся во времени составляющие: аппаратную, операционную и пользовательскую. Последняя составляющая является наиболее неопределенной. Для обеспечения эффективного сопровождения ПС в ней должны быть учтены потенциальные изменения квалификации и субъективных свойств пользователей по мере освоения функциональных возможностей КС, смена и различия персонала, применяющего и использующего КС. Именно то обстоятельство, что КС должна рассматриваться как социотехническая система [29] и обуславливает большую сложность адекватного выбора характеристик, влияющих на безопасность функционирования ПС.

Общими принципами обеспечения безопасности функционирования ПС являются следующие:

1) разрабатываемая система защиты ПС должна быть многоуровневой и ориентированной на все виды угроз с учетом их опасности для потребителя;

2) должна быть обеспечена безопасность функционирования каждого компонента ПС с учетом его уязвимости и степени влияния на безопасность функционирования всего комплекса ПС;

3) разрабатываемая система защиты ПС не должна приводить к помехам и снижению эффективности решения основных задач пользователя;

4) должно быть обеспечено приемлемое соотношение между стоимостью системы защиты ПС и размерами недопустимого для потребителя ущерба.

Эти принципы детализируются с учетом особенностей конкретного проекта.

Выделяют следующие категории отказов, определяющие безопасность функционирования ПС при нарушении его работоспособности:

1) катастрофический отказ, т.е. отказ, при котором может быть нанесен недопустимый по последствиям и величине ущерб КС, объекту управления или пользователям;

2) критический отказ, характеризующийся возможностью возникновения эксплуатационного режима, с которым персонал не может самостоятельно справиться, и который может вызвать неточное или неполное выполнение КС своих функций из-за чего могут проявиться неблагоприятные или потенциально опасные воздействия на окружающую внешнюю среду;

3) существенный отказ, т.е. отказ, при котором снижается функциональная пригодность КС и сокращаются возможности персонала справиться с неблагоприятным эксплуатационным режимом;

4) несущественный отказ, т.е. отказ, при котором незначительно снижается безопасность управления объектом, а персонал самостоятельно может восстановить нормальную работоспособность КС;

5) не влияющий отказ, т.е. отказ, практически не воздействующий на объект управления и не увеличивающий нагрузку на персонал.

Современные стандарты рекомендуют устанавливать уровни безопасности ПС в соответствии с указанными категориями отказов с учетом вероятности возникновения сбоев в программах и данных, выявляемых средствами оценки безопасности КС.

В настоящее время руководствуются следующей классификацией внешних и внутренних угроз безопасности функционирования ПС [26 ,30, 31].

Основными внешними угрозами являются:

1) предумышленное уничтожение, искажение или хищение программ, данных и документации;

2) ошибки и несанкционированные действия оперативного, обслуживающего и административного персонала в процессе эксплуатации КС;

3) недопустимые изменения информации, поступающей от внешних объектов;

4) сбои и отказы аппаратуры КС;

5) деструктивные программы, распространяемые по каналам телекоммуникации;

6) изменения состава и конфигурации КС, а также изменения ПС, выводящие за пределы, проверенные при испытаниях или сертификации.

Основными внутренними угрозами являются:

1) системные ошибки при постановке целей и задач проектирования системы, обеспечивающей безопасность функционирования ПС;

2) ошибки при определении параметров внешней среды применения КС;

3) алгоритмические ошибки при проектировании функций обеспечения безопасности ПС и использовании информации БД;

4) ошибки и дефекты в текстах программ и документации;

5) недостаточная эффективность методов и средств оперативной защиты программ и данных в условиях действия дестабилизирующих факторов.

Отметим, что полное устранение всех перечисленных выше угроз в принципе неосуществимо.

Распределение ресурсов, предназначенных для создания системы защиты ПС, состоит в выделении внешней и внутренней памяти, предназначенной для обеспечения программной и информационной избыточности, для эффективного использования которой должна быть обеспечена временная избыточность, т.е. выделена часть производительности КС [32]. Эти ресурсы должны обеспечивать:

1) контроль и корректировку информации, поступающей от внешней среды и источников данных;

2) средства on-line контроля дефектов исполнения программ и обработки данных;

3) средства защиты от угроз безопасности КС;

4) процедуры мониторинга выявленных дефектов и оперативного восстановления вычислительного процесса, программ и данных после обнаружения дефектов и отказов функционирования КС.

Выбор системы защиты ПС, по своей сути, определяет способ ее установки, а также механизмы и методы защиты ПС [33].

С позиции способа установки выделяют следующие три типа систем защиты ПС:

1. Системы, устанавливаемые на скомпилированные модули ПС после завершения процесса их тестирования. Их достоинство – простота процесса установки. Основной их недостаток – их легко обходить (для этого достаточно определить точку завершения работы системы защиты, передать управления защищенной программе, а затем принудительно ее сохранить в незащищенном виде).

2. Системы, встраиваемые в исходный код ПС до компиляции. Такие системы существенно усложняют процесс тестирования ПС, так как кроме ошибок в ПС могут быть ошибки в системе защиты или в процедурах, использующих систему защиты. Однако они являются более стойкими к атакам, так как исчезает четкая граница между системой защиты и ПС.

3. Комбинированные системы. Такие системы сохраняют достоинства и недостатки систем 2-го типа, но существенно затрудняют анализ и дезактивацию своих алгоритмов.

С позиции используемых механизмов защиты выделяют следующие три типа систем защиты ПС:

1. Системы, использующие сложные логические схемы, ориентированные на затруднение дизассемблирования, отладки и анализа системы защиты и ПС. Основной их недостаток – достаточно низкая стойкость к атакам, так как для преодоления защиты достаточно проанализировать логику процедур проверки и модифицировать их соответствующим образом.

2. Системы, использующие шифрование ПС. Их стойкость к атакам определяется сложностью дешифрации ключа шифрования ПС.

3. Комбинированные системы. Такие системы являются наиболее стойкими к атакам, так как они сохраняют достоинства систем 1-го и 2-го типов.

Основными типами методов защиты ПС являются следующие:

1) запутывание (obfuscation), т.е. использование холостых циклов и хаотических переходов в разные части кода, внедрение ложных процедур (“пустышек”), искажение количества реальных параметров процедур, разброс кода по разным областям памяти и т.д.;

2) мутация, т.е. использование таблиц соответствия операндов-синонимов и замена их друг на друга при каждом запуске программы по определенной схеме или случайным образом;

3) компрессия, т.е. сохранение программы в упакованном виде и ее распаковка только при исполнении;

4) шифрование, т.е. сохранение программы в зашифрованном виде и ее расшифрование только при исполнении;

5) построение логики защиты с использованием элементов, зависящих от результата вычисления значений некоторых формул;

6) использование способов затруднения дизассемблирования в пакетном режиме;

7) применение приемов усложнения отладки программ;

8) эмуляция, т.е. использование виртуального процессора и/или ОС и программы-переводчика системы команд виртуального компьютера и/или ОС в систему команд штатного процессора и/или ОС;

9) применение недокументированных возможностей системы защиты, а также непосредственного обращения модулей системы защиты к аппаратным средствам, минуя процедуры ОС.

Разработку и реализацию системы защиты ПС осуществляет команда, в которую, обычно, входят:

1) менеджер безопасности проекта, обеспечивающий взаимодействие между заказчиком и командой, основная задача которого состоит в определении и удовлетворении требований заказчика по безопасности КС и ПС;

2) архитектор системы защиты ПС, осуществляющий координацию создания ее компонент, разработку базовых функциональных спецификаций, ведение графика ее создания и инициацию принятия возникающих критических решений;

3) специалисты, подготавливающие описания функций компонент системы защиты ПС с детализацией, достаточной для корректной разработки текстов программ;

4) программисты, создающие компоненты системы защиты ПС, удовлетворяющие спецификациям;

5) системные интерпретаторы, создающие требуемые комплексы программ безопасности;

6) специалисты, обеспечивающие проверку функциональных спецификаций и осуществляющие тестирование на каждой фазе процесса создания системы безопасности ПС;

7) специалисты, осуществляющие подготовку и издание сводных технологических и эксплуатационных документов в соответствии со стандартами.

В процессе эксплуатации КС обеспечение жизненного цикла системы безопасности ПС, в основном, осуществляют специалисты, управляющие сопровождением и конфигурацией ПС.

2. Модели и методы верификации ПС

Верификация ПС предназначена для выявления дефектов набора требований, организационных и проектных решений, исходного кода, всей системы ПС и документации к ней с целью приведения их в соответствие со стандартами, обеспечивающими требуемое качество. Эти составляющие процесса верификации ПС характеризуются следующим образом [24, 34].

Верификация набора требований состоит в его анализе на непротиворечивость, полноту, несводимость требований друг к другу, однозначность понимания требований и возможность конструктивной проверки выполнения каждого требования.

Верификация организационных решений состоит в проверке соответствия выбранных форм организации команды разработчиков, планов и методов выполнения работ задачам, решаемым в рамках проекта и ограничениям по срокам и бюджету.

Верификация проектных решений состоит в проверке полноты отражения всех требований в них, анализе непротиворечивости и корректности проектных решений, а также в проверке того, что проектные документы точно и полно формулируют принятые решения.

Верификация исходного кода состоит в проверке того, что он написан в соответствии со стандартно оформленными текстами программ, выполнены требования к удобству его сопровождения, корректно реализованы проектные решения, отсутствуют непредусмотренные требованиями действия, а также отсутствуют пути, приводящие к сбоям, за-

цикливанию, тупиковым ситуациям, разрушению процессов и данных.

Верификация системы ПС состоит в проверке ее способности эффективно функционировать в окружении, определенном требованиями заказчика.

Верификация документации состоит в проверке полноты, точности и непротиворечивости описания поведения системы ПС, а также соответствия описания реальному поведению системы ПС.

В настоящее время выделяют следующие группы методов верификации ПС:

1) экспертиза, т.е. осуществляемый специалистами критический анализ системы ПС в контексте ее жизненного цикла;

2) статический анализ, т.е. проверка корректности формализованных правил построения ПС и поиск часто встречающихся дефектов по шаблонам;

3) формальные методы, т.е. анализ свойств ПС на основе формальных моделей требований, поведения ПС и их окружения;

4) динамический анализ, т.е. проверка и оценка свойств ПС на основе их реальной работы;

5) синтетические методы, т.е. методы, построенные на основе комбинаций методов из перечисленных выше групп.

Эти группы методов верификации характеризуются следующим образом.

Экспертизы ПС, как показывает практика, при относительно небольших затратах на их проведение дают возможность выявить 50-90% ошибок, зафиксированных на протяжении жизненного цикла ПС.

Первая, успешно примененная на практике, технология проведения экспертизы ПС – оценка ПС по Фагану (Fagan software inspection) – разработана в 1972 г. [35]. Она представляет собой четко структурированную техническую экспертизу ПС и сопровождающей документации, управление исполнения которой основано на использовании сквозного контроля. Термин «техническая экспертиза» означает систематический анализ проекта специалистами с целью оценки его корректности, точности, полноты, соответствия стандартам и поставленным задачам (если техническая экспертиза выполняется специалистами, не входящими в команду проекта, то она называется аудитом). Термин «сквозной контроль» означает brainstorming, организованный следующим образом: один из участников последовательно представляет остальным установленные им характеристики анализируемого фрагмента, а они методом диалога вносят свои предложения, указывают на возможные дефекты и нарушения стандартов.

Последующие технологии проведения экспертизы ПС [36-38] основаны на развитии технологии оценки ПС по Фагану и расширении множества объектов, подвергаемых экспертизе. По-видимому, основными являются следующие достижения в этом направлении. Во-первых, это использование на различных этапах экспертизы инспекций, поддерживаемых инструментальными средствами, т.е. проверок, в ходе которых в соответствии с заданным списком проверяется наличие указанных в нем свойств и отсутствие указанных в нем дефектов. Во-вторых, это разработка техники чтения исходного кода с постепенным обобщением (reading by stepwise abstraction), состоящей в последовательном построении и уточнении реализуемой им функции [39]. В-третьих, это разработка методов инспекции пользовательского интерфейса [40, 41], методов экспертизы (аудита) защищенности ПС [42, 43] и методов экспертизы архитектуры ПС [44].

Статический анализ применяется для верификации исходного кода и его архитектуры [45]. Установленные на практике правила построения корректного исходного кода и шаблоны типичных ошибок переносятся в среды разработки ПС (Eclipse, Microsoft Visual Studio) и среды моделирования их архитектуры (Rational Rose). Такие правила трансформируются в семантические правила соответствующих языков программирования, а проверка их выполнения осуществляется компиляторами этих языков.

Формальные методы верификации ПС основаны на представлении требований, поведения и окружения ПС логико-алгебраическими моделями и абстрактными машинами. Эти типы моделей характеризуются следующим образом.

Логические модели, применяемые при верификации ПС [46, 47], основаны на использовании теорий 1-го порядка и предназначены для анализа истинности, ложности или выполнимости формул соответствующего языка. Разработка методов анализа формул таких теорий и привела к созданию инструментальных средств автоматизированного решения (т.е. решателей) ряда задач верификации ПС [48, 49]. Построение соответствующих теорий 1-го порядка основано на использовании исчислений высказываний и предикатов [50, 51], λ-исчисления [52] и модальных логик [53]. Исчисление предикатов предоставляет средства конструктивного построения формул только при соблюдении соответствия типов параметров типам подставляемых вместо них выражений, λ-исчисление предоставляет средства конструктивного построения функций из выражений, а модальные логики предоставляют средства конструктивного построения утверждений со смысловой нагрузкой. Наиболее известными детализациями модальной логики являются временные логики, предоставляющие средства описания последовательностей событий во времени.

Следует подчеркнуть, что принципиальные ограничения применения классических теорий при разработке ПС привели к разработке композиционно подхода в программировании [54], важной составляющей которого являются исчисления квазиарных предикатов [55], т.е. частичных отображений, заданных на произвольных наборах именованных значений. По-видимому, именно такие исчисления предикатов и являются теоретической основой построения решателей, предназначенных для верификации ПС.

Алгебраические модели предназначены для анализа выражений или термов того или иного вида. Отметим, что различие между логическими и алгебраическими моделями достаточно условное, так как для любой алгебраической системы всегда можно построить эквивалентное ей логическое типизированное исчисление [56].

Алгебраическими моделями, применяемыми при верификации ПС, являются реляционные (табличные) алгебры [57], алгебраические модели абстрактных типов данных [24] и алгебры процессов [58, 59].

Реляционные алгебры представляют собой теоретическую основу построения систем управления реляционными базами данных. Систематическому анализу теоретико-множественных конструкций, лежащих в основе реляционных алгебр, а также обобщениям этих алгебр на бесконечные таблицы и мультимножества посвящены работы [60-63].

Алгебраические модели абстрактных типов данных предназначены для аксиоматического построения и формального анализа основных типов структур данных [64-67], а также для формального анализа отношения номинации (именования) в языках спецификации программ [68-69].

Алгебры процессов моделируют взаимодействия процессов, являющихся математическими моделями исполняемых программ, в терминах обмена порождаемых ими сообщений (порождаемых в свою очередь событиями). Отметим, что алгебры и исчисления процессов применимы для верификации мобильных вычислений (т.е. вычислений в динамически изменяющихся сетях, в которых осуществляется перенос вычислительных процессов из одного узла в другой) [70, 71].

Общей характеристикой всех рассмотренных выше моделей верификации ПС является то, что они являются моделями дескриптивного типа, т.е. их применение состоит в анализе соотношений, представленных на соответствующем языке.

Абстрактные машины предназначены для верификации тех характеристик анализируемой ПС, которые моделируются «программой» этой машины, т.е. любая абстрактная машина – исполнимая модель. Наиболее часто для верификации ПС применяются конечные автоматы [72], системы помеченных переходов (labeled transition systems) [73], сети Петри [74, 75] и машины абстрактных состояний [76, 77].

Стоит выделить следующие детализации конечных автоматов, предназначенных для моделирования ПС. Во-первых, это расширенные конечные автоматы [78], т.е. конечные автоматы, в которых каждый переход осуществляется только при выполнении соответствующего «охранного условия» (guard condition). Во-вторых, это временные автоматы [79], т.е. расширенные конечные автоматы, снабженные переменными-таймерами, которые используются в условиях, определяющих переходы. В-третьих, это гибридные автоматы [80], т.е. автоматы, предназначенные для моделирования взаимодействия ПС с непрерывными процессами (в таких автоматах динамика некоторого множества переменных представлена системой дифференциальных уравнений).

Отметим, что системы помеченных переходов являются исполнимым аналогом алгебры процессов, сети Петри предоставляют возможность явного описания параллелизма процессов, а в машинах абстрактных состояний переходы представляются «программой», состоящей из инструкции, переопределяющей значение функционального символа, инструкций типа «условие» и инструкций параллельного выполнения действий для всех объектов, удовлетворяющих тому или иному условию.

По-видимому, перспективным средством унифицированной разработки моделей абстрактных машин, предназначенных для верификации ПС, является визуальная технология программирования на основе Р-схем [81-83]. Для этой технологии в настоящее время реализованы графический редактор, транслятор Р-схем в С++ и система проектирования Р-схем. Безусловным достоинством этой технологии является ее наглядность, компактность представления программ, а также возможность иерархического представления программ, так как каждая вершина графа, представляющего данную программу, может быть рассматриваться как имя графа, представляющего некоторую программу.

Основными методами построения доказательства того, что формальная модель удовлетворяет спецификации, являются проверка модели (model checking) [47] и логический вывод [84]. Проверка модели представляет собой автоматический анализ заданной в явном или в неявном виде формальной модели, результатом которого в случае, когда модель не удовлетворяет спецификации, является «опровергающее вычисление», т.е. последовательности действий, на которых нарушается специфика-

ция. Логический вывод состоит в построении и формальном доказательстве утверждений (инвариантов), которые истинны в каждый момент времени и из конъюнкции которых вытекает спецификация модели.

Динамический анализ ПС применяется для проверки соответствия результатов реальной работы проверяемой ПС требованиям и проектным решениям. Подчеркнем, что динамический анализ ПС дает возможность исследовать такие основные характеристики ПС, как функциональность, производительность, переносимость, надежность и удобство использования. Динамический анализ может быть как имитационным (если при его реализации используется исполнимая модель ПС), так и реальным (если при его реализации используется сама ПС).

Методы динамического анализа ПС делятся на мониторинг и тестирование.

Мониторинг ПС представляет собой наблюдение, фиксацию и оценку функционирования ПС в процессе ее обычного функционирования. В настоящее время разработано достаточно много промышленных средств мониторинга, либо входящих в тот или иной инструментарий разработки ПС, либо распространяемых отдельно.

Тестирование ПС представляет собой наблюдение, фиксацию и оценку функционирования ПС в ситуациях, определяемых заранее подготовленными сценариями. Полнота тестирования определяется процентом покрытых им классов ситуаций, а адекватность тестирования – критериями, построенными на основе анализа рисков проекта с учетом приоритета ошибок. Наиболее распространенными методами построения тестовых последовательностей являются вероятностное тестирование [85], тестирование на основе классов эквивалентности [86-88] и комбинированное (combination) тестирование [89].

Подчеркнем, что тестирование ПС применимо и для валидации ПС. В этом случае для оценки соответствия ПС требованиям заказчика привлекаются пользователи ПС, а также эксперты в данной предметной области.

3. Безопасность БД

Любая БД представляет собой такую организацию хранения взаимосвязанных данных, которая за счет заложенной в ней избыточности допускает эффективное использование данных для одного или нескольких приложений. Таким образом, любая БД является частью соответствующего ПС, существующей независимо от конкретных исполнимых программ и, как правило, предназначенной для совместного использования достаточно большим числом пользователей. Именно эта особенность БД и потребовала создания соответствующих систем управления базами данных (СУБД), т.е. ПС, обеспечивающих корректное размещение данных, надежное их хранения, а также поиск, модификацию и удаление данных. В настоящее время используются различные модели данных, основными из которых являются иерархическая, сетевая, реляционная, объектно-ориентированная, объектно-реляционная и многомерная модели. Такое многообразие моделей данных естественно приводит к различным СУБД, ориентированных на соответствующую модель данных.

Основные требования по безопасности БД и СУБД во многом совпадают с требованиями, предъявляемыми к безопасности ПС. Кроме того, возникает необходимость обеспечения управления целостностью, как элементов данных, так и логических взаимосвязей между ними. Нарушение такой целостности может быть вызвано следующими дестабилизирующими факторами:

1) сбоями оборудования, физическими действиями или стихийными бедствиями;

2) ошибками ОС, СУБД или прикладных программ;

3) ошибками или неквалифицированными действиями (легальных) пользователей;

4) возникновением конфликтов из-за действий (легальных) пользователей;

5) несанкционированными действиями (легальных) пользователей;

6) действиями нелегальных пользователей или вредоносных программ.

Для предотвращения действия большинства из этих дестабилизирующих факторов предназначены следующие средства СУБД.

Во-первых, современные СУБД, как правило, содержат встроенные средства, которые дают возможность администратору определять для каждого пользователя его права доступа к тем или иным частям БД (вплоть до элемента), а также тип доступа, т.е. те действия с БД, которые разрешены данному пользователю. При этом основная проблема, возникающая на практике, состоит в обеспечении своевременного управления доступом пользователей.

Во-вторых, это средства СУБД, обеспечивающие устойчивость элементов данных и их логических взаимосвязей к ошибкам и/или неквалифицированным действиям пользователей. В основе построения таких средств лежит следующий принцип: в каждый элемент данных информация заносится точно в соответствии с описанием этого элемента.

В-третьих, это средства, обеспечивающие поддержание целостности БД при возникновении конфликтов. Эта проблема имеет большую внутреннюю сложность, так как нарушение целостности БД мо-

жет возникать даже при одновременном выполнении нескольких операций, каждая из которых не нарушает целостности БД. Предотвращение таких ситуаций основано на объединении в один логический элемент функционирования БД, называемый транзакцией, тех операций, в результате которых БД из одного целостного состояния переходит в другое целостное состояние.

Применение транзакции состоит в том, что до ее завершения все действия с данными проводятся вне БД, а занесение реальных изменений в БД производится только после нормального ее завершения. При этом копия результатов выполнения транзакции записывается в системный журнал (такая операция называется фиксацией).

Для того чтобы избежать ситуации, когда в результате одновременного выполнения нескольких транзакций, каждая из которых не нарушает целостность БД, может быть нарушена целостность БД, современные СУБД содержат средства, обеспечивающие захват транзакцией модифицируемых элементов данных до момента завершения модификации. Такие средства называются блокировками. Применение блокировки состоит в том, что никакая иная транзакция не получит доступа к модифицируемому элементу данных до тех пор, пока данная транзакция не освободит его.

В процессе применении блокировок может возникать ситуация, когда некоторая транзакция пытается блокировать объект, который уже блокирован другой транзакцией. В этом случае ей приходится ожидать снятия блокировки объекта транзакцией, установившей эту блокировку. В результате может возникнуть достаточно сложная проблема управления очередями транзакций. Отметим, что ее эффективное решение существенно влияет на эффективность использования БД.

Другая проблема возникает при прекращении выполнения транзакции вследствие появления ошибки в программе или того или иного сбоя. Для ее решения в современных СУБД содержатся следующие встроенные средства:

1. Средства, осуществляющие отмену прерванной транзакции и возврат БД в состояние, предшествующее началу ее выполнения. В этом случае до устранения неисправностей, вызванных сбоем, применяется монопольная блокировка всех транзакций.

2. Средства, осуществляющие возврат прерванной транзакции в некоторую промежуточную ее точку (такая точка называется точкой фиксации или контрольной точкой). В процессе выполнения транзакции пользователь может установить произвольное количество таких точек. Если в ходе выполнения транзакции достигается точка фиксации, то СУБД автоматически осуществляет возврат в эту точку. Для этого с помощью системного журнала восстанавливаются значения проведенных изменений в порядке их возникновения; такая операция называется раскруткой транзакции. Хотя раскрутка транзакции является достаточно сложной операцией, именно эта операция существенно влияет на эффективность использования БД.

В-четвертых, это средства, которые при возникновении неисправности восстанавливают состояние БД, которое было перед ее появлением. Выделяют следующие три уровня восстановления БД:

1) оперативное восстановление, т.е. восстановление на уровне отдельной транзакции;

2) промежуточное восстановление, т.е. восстановление состояний всех транзакций, выполняемых на момент возникновения системно-программных ошибок или сбоя программного обеспечения, не связанные с разрушением БД;

3) длительное восстановление, т.е. восстановление БД при ее разрушении.

Оперативное и промежуточное восстановление основаны на использовании операции раскрутки транзакции. Длительное восстановление осуществляется с помощью копии БД и воспроизведения результатов транзакций, выполненных с момента снятия копии, в результате чего восстанавливается состояние БД на момент разрушения.

Заключение

В работе кратко рассмотрены модели и методы, применяемые для решения сложной многогранной проблемы обеспечения безопасности функционирования современных ПС на протяжении всего их жизненного цикла. Существующие в настоящее время подходы к решению этой проблемы весьма далеки от того, что принято называть проработанной технологией. Отметим некоторые задачи, решение которых необходимо для создания указанной технологии.

Во-первых, это задача разработки таких эффективных методов валидации ПС на различных этапах их проектирования и разработки, которые, в своей совокупности, исключат неоправданное удлинение срока выполнения проекта.

Во-вторых, это задача разработки эффективных автоматизированных средств верификации логико-алгебраических моделей и абстрактных машин. Достижение этой цели требует, прежде всего, теоретического исследования задач выполнимости формул соответствующих теорий.

В-третьих, это задача разработки эффективных средств, предназначенных для противодействия несанкционированному исследованию ПС. Достижение этой цели требует системного анализа соотно-

шения между современными средствами противодействия несанкционированному исследованию ПС и методами их преодоления (см., напр., [90-92]). В частности, требуют внимания системные теоретические и прикладные исследования методов запутывания программ (см., напр., [93]).

Литература

1. *Казарин О.В. Безопасность программного обеспечения компьютерных систем [Текст] / О.В. Казарин. – М: МГУЛ, 2003. – 212 с.*
2. *Боэм Б. Характеристики качества программного обеспечения [Текст] / Б. Боэм, Х. Каспар. – М.: Мир, 1981. – 208 с.*
3. *Боэм Б. Инженерное проектирование программного обеспечения [Текст] / Б. Боэм. – М.: Радио и связь, 1985. – 512 с.*
4. *Brooks F.P. No silver bullet – essence and accidents of software engineering [Текст] / F.P. Brooks // Proc. of the IFIP 10th World Computing Conference – 1986. – P. 1069-1076.*
5. *Miller H. Scoping the global market: size is just part of the story [Текст] / H. Miller, J. Sanders // IT Professional. – 1999. – № 1. – P. 49-54.*
6. *Юсупов Р. М. Безопасность компьютерной инфосферы систем критических приложений / Р. М. Юсупов, Б. П. Пальчун // Вооружение. Политика. Конверсия. – 1993, № 2. – С. 52-56.*
7. *Федотов А.М. Информационная безопасность в корпоративной сети / А.М. Федотов // Проблемы безопасности и чрезвычайных ситуаций. – М.: ВИНИТИ, 2008. – № 2. – С. 88-101.*
8. *Castano S. Database security [Текст] / S. Castano, M.G. Fugini, G. Martello, et al. – N.Y.: Addison Wesley Publishing Company, 1995. – 456 p.*
9. *Гайдамакин Н.А. Теоретические основы компьютерной безопасности [Текст] / Н.А. Гайдамакин. – Екатеринбург: Изд-во Уральского университета, 2008. – 212 с.*
10. *Щербаков А.Ю. Современная компьютерная безопасность. Теоретические основы. Практические аспекты [Текст] / А.Ю. Щербаков. – М.: Книжный мир, 2009. – 352 с.*
11. *Ухлинов Л.М. Защита данных в информационно-вычислительных сетях: обзор технологий [Текст] / Л.М Ухлинов // Вестник РОИВТ. – 1992. – № 1-2. – С. 39.*
12. *Юсупов Р.М. Обеспечение безопасности компьютерной инфосферы [Текст] / Р.М. Юсупов, В.П. Пальнув // Вооружение. Политика. Конверсия. – 1993. – № 3. – С. 23.*
13. *Зегжда Д.П. Проблема анализа безопасности программного обеспечения [Текст] / Д.П. Зегжда, Э.М. Шмаков // Безопасность информационных технологий. – 1995. – № 2. – С. 28-33.*
14. *Казарин О.В. О создании информационных технологий, исходно ориентированных на разработку безопасного программного обеспечения [Текст] / О.В. Казарин // Безопасность информационных технологий. – 1997. – № 1-2. – С. 9-10.*
15. *Зегжда Д.П. Основы безопасности информационных систем [Текст] / Д.П. Зегжда, А.М. Ивашко. – М.: Горячая линия-Телеком, 2000. – 452 с.*
16. *Шаньгин В.Ф. Информационная безопасность компьютерных систем и сетей [Текст] / В.Ф. Шаньгин. – М.: ИД «ФОРУМ», 2011. – 416 с.*
17. *Microsoft solutions framework. [Электронный ресурс]. – Режим доступа: http://www.microsoft.com/Rus/Msdn/msf/Default.mspx.*
18. *Rational Unified Process. Методология и технология. Материалы компании Interface Ltd [Электронный ресурс]. – Режим доступа: http://www.interface.ru/home.asp?artId=779.*
19. *Бек К. Экстремальное программирование [Текст] / К.Бек. – СП-б: Питер, 2002. – 224 с.*
20. *Скобелев В.Г. Безопасность IT-систем (обзор) [Текст] / В.Г. Скобелев // Радіоелектронні і комп'ютерні системи. – 2013. – № 5. – С. 352-361.*
21. *IEEE 610.12-1990 Standard glossary of software engineering terminology, corrected edition [Текст]. – IEEE, 1991.*
22. *IEEE 1012-2004 Standard for verification and validation [Текст]. – IEEE, 2005.*
23. *ISO/IEC 12207 Systems and software engineering – software life cycle processes [Текст]. – ISO, 2008.*
24. *Кулямин В.В. Методы верификации программного обеспечения [Электронный ресурс] / В.В. Кулямин. – Режим доступа: http://www/viva64.com/go.php?url=282.*
25. *Howden W.E. Functional program testing and analysis [Текст] / W.E. Howden, – N.Y.: McGraw Hill, 1987. – 175 p.*
26. *Галатенко В.А. Основы информационной безопасности [Текст] / В.А. Галатенко. – М.: ИНТУИТ, 2003. – 208 с.*
27. *Липаев В.В. Технологические процессы и стандарты обеспечения функциональной безопасности в жизненном цикле программных средств [Текст] / В.В. Липаев // Jet Info. – 2004. – № 3. – С. 2-19 .*
28. *Липаев В.В. Функциональная безопасность программных средств [Текст] / В.В. Липаев // Jet Info. – 2004. – № 8. – С. 3-28.*
29. *Волобуев С.В. Философия безопасности социотехнических систем: информационные аспекты [Текст] / С.В. Волобуев. – М.: Вузовская книга, 2004. – 360 с.*
30. *Устинов Г.Н. Основы информационной безопасности систем и сетей данных [Текст] / Г.Н. Устинов. – М.: СИНТЕГ, 2000. – 248 с.*
31. *Фатрелл Р.Т. Управление программными проектами: достижение оптимального качества при минимуме затрат [Текст] / Р.Т. Фатрелл, Д.Ф. Шафер, Л.И. Шафер. – М.: Вильямс, 2003. – 1136 с.*

32. Липаев В.В. Распределение ресурсов в вычислительных системах / В.В. Липаев. – М.: Статистика, 1979. –247 с.

33. Середа С.А. Оценка эффективности систем защиты программного обеспечения [Электронный ресурс] / С.А. Середа. – Режим доступа: http://consumer.nm.ru/sps_eval.htm.

34. Синицин С.В. Верификация программного обеспечения [Текст] / С.В. Синицин, Н.Ю. Налютин. – М.: МИФИ, 2006. – 157 с.

35. Fagan M.E. Design and code inspections to reduce errors in program development [Текст] / M.E. Fagan // IBM Systems Journal. – 1976. – N 3. – P. 182-211.

36. Gilb T. Software inspection [Текст] / T. Gilb, D. Graham. – NY: Addison-Wesley, 1993. – 471 p.

37. Laitenberger O. A survey of software inspections technologies [Текст] / O. Laitenberger // In: Handbook on software engineering and knowledge engineering. Vol. 2. – NY: World Scientific Publishing, 2002. – P. 517-555.

38. Wong Y.K. Modern software review [Текст] / Y.K. Wong. – NY: IRM Press, 2006. – 324 p.

39. Dyer M. Verification-based inspection [Текст] / M. Dyer // Proc. of 26th Annual Hawaii International Conference on Systems Sciences. – 1992. – P. 418-427.

40. Nielsen J. Usability engineering [Текст] / J. Nielsen. – Boston: Academic Press, 1993. – 363 p.

41. Константайн Л. Разработка программного обеспечения [Текст] / Л. Константайн, Л. Локвуд. – СПб: Питер, 2004. – 592 с.

42. Anderson R. Security engineering: a guide to building dependable distributed systems [Текст] / R. Anderson. – NY: John Wiley & Sons, 2001. – 1040 p.

43. Шнайер Б. Секреты и ложь: безопасность данных в цифровом мире [Текст] / Б. Шнайер. – СПб: Питер, 2003. – 368 с.

44. Dobrica L. A survey on software architecture analysis methods [Текст] / L. Dobrica, E. Niemela // IEEE Transactions on software engineering. – 2002. – № 7. – P. 638-653.

45. Yu L. A light-weight static approach to analyzing UML behavioral properties [Текст] / L. Yu, R.B. France, I. Ray, R. Lano // Proc. of 12th IEEE International Conference on Engineering Complex Systems. – 2007. – P. 56-63.

46. Ин К. О программных логиках – просто [Текст] / К. Ин, Н.В. Шилов, Е.В. Бодин // Системная информатика. – 2002. – № 8. – С. 206-249.

47. Кларк Э. Верификация моделей программ: model checking [Текст] / Э. Кларк, О. Грамберг, Д. Пелед. – М.: МЦНМО, 2002. – 416 с.

48. Летичевский А.А. Алгоритмы очевидности Глушкова [Текст] / А.А. Летичевский, А.В. Лялецкий, М.К. Мороховец // Кибернетика и системный анализ. – 2013. – № 4. – С. 3-16.

49. Скобелев В.В. Проблема проверки выполнимости формул разрешимых теорий (обзор) [Текст] / В.В. Скобелев // Труды ИПММ НАНУ. – 2013. – Т. 26. – С. 205-221.

50. Мендельсон Э. Введение в математическую логику [Текст] / Э. Мендельсон. – М.: Наука, 1971. –320 с.

51. Ершов Ю.Л. Математическая логика [Текст] / Ю.Л. Ершов, Е.А. Палютин. – СПб: Лань, 2004. – 336 с.

52. Барендрег Х. Лямбда-исчисление. Его синтаксис и семантика [Текст] / Х. Барендрег. – М.: Мир, 1985. – 606 с.

53. Фейс Р. Модальная логика [Текст] / Р. Фейс. – М.: Наука, 1974. – 516 с.

54. Редько В.Н. Основания композиционного программирования [Текст] / В.Н. Редько // Программирование. – 1979. – № 3. – С. 3-13.

55. Нікітченко М.С., Шкільняк С.С. Математична логіка та теорія алгоритмів [Текст] / М.С. Нікітченко, С.С. Шкільняк. – Київ: ВПЦ "Київський університет", 2008. – 528 с.

56. Мальцев А.И. Алгебраические системы [Текст] / А.И. Мальцев. – М.: Наука, 1970. – 329 с.

57. Дейт Дж. Введение в системы баз данных [Текст] / Дж. Дейт. – М.: Вильямс, 2005. – 1328 с.

58. Котов В.Е. Исчисления процессов [Текст] / В.Е. Котов, Л.А. Черкасова // Системная информатика. – 1993. – № 2. – С. 6-38.

59. Хоар Ч. Взаимодействующие последовательные процессы [Текст] / Ч. Хоар. – М.: Мир, 1989. – 264 с.

60. Буй Д.Б. Теоретико-множинні конструкції в теорії реляційних баз даних [Текст] / Д.Б. Буй, Ю.Й. Брона // Вісник Київського національного університету. Сер.: фіз.-мат. науки. – 1996. – Вип. 1. – С. 216-224.

61. Редько В.Н. К основаниям теории реляционных моделей баз данных [Текст] / В.Н. Редько, Д.Б. Буй // Кибернетика и системный анализ. – 1996. – № 4. – С. 3-12.

62. Редько В.Н. Реляційні бази даних: табличні алгебри та SQL-подібні мови [Текст] / В.Н. Редько, Ю.Й. Брона, Д.Б. Буй, С.А. Поляков. – Київ: Академперіодика, 2001. – 198 с.

63. Буй Д.Б. Узагальнена таблична алгебра та узагальнене реляційне обчислення [Текст] / Д.Б. Буй, І.М. Глушко // Вісник Київського національного університету. Сер.: фіз.-мат. науки. – 2011. – Вип. 1. – С. 86-95.

64. Oppen D.C. Reasoning about recursively defined data structures [Текст] / D.C. Oppen // Journal of the ACM. – 1980. – N 3. – P. 403-411.

65. Bjorner N. Deciding fixed and non-fixed size bit vectors [Текст] / N. Bjorner N. // LNCS. – 1998. – Vol. 1384. – P. 376-392.

66. Stump A. A decision procedure for for an extensional theory of arrays [Текст] / A. Stump, D.L. Dill, C.W. Barret at all. // Proc. of LICS'01. – 2001. – P. 29-37.

67. Manna Z. Combining decision procedures [Текст] / Z. Manna, C. Zarba C // LNCS. – 2003. – Vol. 2787. – P. 453-468.

68. Россада Т.В. Формалізація багатозначного іменування у мовах програм над номінативними даними [Текст] / Т.В. Россада // Вісник Київського національного університету. Сер.: фіз.-мат. науки. – 2012. – Вип. 1. – С. 218-227.

69. Россада Т.В. Властивості слабкоструктурованих даних з багатозначним іменуванням та їх використання [Текст] / Т.В. Россада, А.В. Скляр // Вісник Київського національного університету. Сер.: фіз.-мат. науки. – 2012. – Вип. 2. – С. 217-222.

70. Milner R. Communicating and mobile systems: the π-calculus [Текст] / R. Milner. – Cambrige University Press, 1999. – 165 p.

71. Cardelli L. Mobile ambient [Текст] / L. Cardelli, A.D. Gordon // LNCS. – 1998. – Vol. 1378. – P. 140-155.

72. Хопкрофт Дж. Введение в теорию автоматов, языков и вычислений [Текст] / Дж. Хопкрофт, Р. Мотвани, Дж. Ульман. – М.: Вильямс, 2002. – 528 с.

73. Кузьмин Е.В. Структурированные системы переходов [Текст] / Е. В. Кузьмин, В.А. Соколов. – М.: Физматлит, 2006. – 176 с.

74. Питерсон Дж. Теория сетей Петри и моделирование систем [Текст] / Дж. Питерсон. – М.: Мир, 1984. – 264 с.

75. Котов В.Е. Сети Петри [Текст] / В.Е. Котов. – М.: Наука, 1984. – 160 с.

76. Gurevich Y. Evolving algebras [Текст] / Y. Gurevich // Bulletin of the European Association for Theoretical Computer Science. – 1991. – Vol. 43. – P. 264-284.

77. Гуревич Ю. Последовательные машины абстрактных состояний охватывают последовательные алгоритмы [Текст] / Ю. Гуревич // Системная информатика. – 2004. – № 9. – С. 7-50.

78. Simon G.A. An extended finite state machine approach to protocol specification [Текст] / G.A. Simon // Proc. of 2nd International Workshop on Protocol Specification, Testing and Verification. – 1982. – P. 113-133.

79. Alur R. A theory of timed automata [Текст] / R. Alur, D.L. Dill // Theoretical Computer Science. – 1994. – Vol. 126. – P. 183-235.

80. Henzinger T.A. The theory of hybrid automata [Текст] / T.A. Henzinger // Proc. of LICS'96. – 1996. – P. 278-292.

81. Глушков В.М. Технология программирования и проблемы ее автоматизации [Текст] / В.М. Глушков, И.В. Вельбицкий // УСИМ. – 1976. – № 6. – С. 75-93.

82. Вельбицкий И.В. Технология программирования [Текст] / И.В. Вельбицкий. – Киев: Техника, 1984. – 279 с.

83. McHenry W.K. R-technology: A Soviet visual programming [Текст] / W.K. McHenry // Journal of Visual Languages and Computing. – 1990. – Vol. 1. – P. 199-212.

84. Eriksson J. Applying PVS background theories and proof strategies in invariant based programming [Текст] / J. Eriksson, R.J. Back // // LNCS. – 2010. – Vol. 6447. – P. 24-39.

85. Hamlet D. Random testing [Текст] / D. Hamlet // In: Encyclopedia of software engineering. – NY: Wiley, 1994. – P. 970-978.

86. Ostrand T.J. The category-partition method for specifying and generating functional tests [Текст] / T.J. Ostrand, M.J. Balcer // Communications of the ACM. – 1988. – № 3. – P. 676-686.

87. Липаев В.В. Тестирование программ [Текст] / В.В. Липаев. – М.: Радио и связь, 1986. – 296 с.

88. Майерс Г. Искусство тестирования программ [Текст] / Г. Майерс. – М.: Финансы и статистика, 1982. – 176 с.

89. Grindal M. Combination testing strategies: a survey [Текст] / M. Grindal, J. Offutt, S. Andler // Software Testing, Verification and Reliability. – 2005. – № 3. – P. 167-199.

90. Расторгуев С.П. Искусство защиты и раздевания программ [Текст] / С.П. Расторгуев, Н.Н. Дмитриевский – М.: Совмаркет, 1991. – 94 с.

91. Семьянов П.В. Анализ средств противодействия исследованию программного обеспечения и методы их преодоления / П.В. Семьянов, Д.П. Зегжда [Электронный ресурс]. – Режим доступа: http://www.password-crackers.com/publications/research.txt.

92. Середа С.А. Анализ средств преодоления систем защиты программного обеспечения [Текст] / С.А. Середа // Радиоэлектроника и Телекоммуникации. – 2002.– № 4 – С. 11-16.

93. Чернов А.В. Анализ запутывающих преобразований программ / А.В. Чернов [Электронный ресурс]. – Режим доступа: http://www.citrorum.ru/security/articles/analysis/.

Поступила в редакцию

Рецензент:

БЕЗПЕКА ПРОГРАМНИХ ЗАСОБІВ: МОДЕЛІ ТА МЕТОДИ (ОГЛЯД)

Д.Б. Буй, В.Г. Скобелєв

В роботі дано короткий огляд моделей та методів, які призначено для забезпечення інформаційної та функціональної безпеки сучасних IT-систем. Охарактеризовано сервери безпеки, методи аналізу та формування політики безпеки, методи безпосереднього тестування, моделі і методи керування доступу та методи забезпечення програмної безпеки IT-систем.

Ключові слова: програмні засоби, системи захисту програмних засобів, верифікація.

SECURITY OF SOFTWARE: MODELS AND METHODS (A SURVEY)

D.B. Bui, V.G. Skobelev

In the given paper it is presented some short survey of models and methods intended to provide security of software in modern IT-systems in the process of their evolution. Existing approaches applied for elaboration of technologies intended to provide security of software and difficulties that arise in this process are considered. General scheme applied for realization of security system for software is characterized. Basic types of security system for software, as well as basic methods applied for their elaboration are presented. Models and methods applied in the process of verification of software are characterized. Some trends of research connected with elaboration of formal models and methods of verification of software are pointed. Methods of database protection are presented shortly.

Keywords: *software, software security systems, verification.*

Буй Дмитрий Борисович – д-р физ.-мат наук, профессор, профессор кафедры теории и технологии программирования факультета кибернетики Киевского национального университета имени Тараса Шевченко, Киев, Украина, e-mail: buy@unicyb.kiev.ua

Скобелев Владимир Геннадиевич – д-р физ.-мат наук, д-р техн. наук, профессор, ведущий научный сотрудник Института прикладной математики и механики НАН Украины, Донецк, Украина, e-mail: skbv@iamm.ac.donetsk.ua