МУЛЬТИАГЕНТНАЯ СИСТЕМА ЗАЩИТЫ РАСПРЕДЕЛЕННОЙ ИМИТАЦИОННОЙ МОДЕЛИ С УДАЛЕННЫМ ДОСТУПОМ

Александр Миков, Елена Замятина, Михаил Панов

***Аннотация**: В работе рассматривается распределенная система имитации Triad.Net с удаленным доступом. Обсуждаются программные средства, позволяющие исследователям удаленно и совместно с другими исследователями через Internet взаимодействовать с имитационной моделью, а также наблюдать за поведением модели во время имитационного эксперимента. Реализация удаленной системы диктует необходимость разработки подсистемы защиты, которая осуществляет обнаружение вторжений и противодействует злоумышленникам. Для реализации подсистемы защиты был выбран мультиагентный подход, позволяющий осуществить эффективную работу в компьютерной сети.*

***Keywords**: Имитационное моделирование; системы защиты от вторжений; мультиагентная система.*

***ACM Classification Keywords**: I.6 Simulation and Modelling – I.6.2 Simulation Languages; I.2 Artificial Intelligence – I.2.5 Programming Languages and Software – Expert system tools and techniques.*

***Conference**: The paper is selected from Seventh International Conference on Information Research and Applications – i.Tech 2009, Varna, Bulgaria, June-July 2009*

Введение

В настоящее время актуален переход к созданию распределенных систем имитационного моделирования (СИМ). Как известно, имитационное моделирование является одним из наиболее часто используемых, а иногда и единственным методом исследования сложных систем. Сложность задач, решаемых методом имитационного моделирования, является причиной разработки такого программного обеспечения, которое позволило бы использовать ресурсы гетерогенных или гомогенных многопроцессорных или мультикомпьютерных вычислительных систем (ВС). Кроме того, актуально и предоставление пользователям средств удалённого доступа к имитационным моделям. Исследователи, которые находятся в разных географических точках, получают возможность работать над одним имитационным проектом, совместно разрабатывать имитационные модели и наблюдать за ходом имитационного эксперимента. Удаленный доступ требует предоставления средств защиты от взломов и внешних атак, проработки вопросов аутентификации, прав доступа для пользователей различных категорий, т.е. разработки специальной подсистемы безопасности. Прежде, чем остановиться на обсуждении вопросов, связанных с проектированием и реализацией подсистемы безопасности, рассмотрим архитектуру распределенной системы имитации.

Распределенная система имитации

Распределенная система имитации разрабатывается с применением технологии .Net. Использование технологии .Net дают возможность реализовать гибкую компонентно-ориентированную систему. Для синхронизации распределенных объектов имитационной модели разработаны консервативный и оптимистический алгоритмы. Triad.Net является распределенной версией системы моделирования Triad. Первая версия этой системы разрабатывалась в 90-х годах и предназначалась для автоматизированного проектирования и моделирования вычислительных систем. Первоначальная цель отразилась на представлении имитационной модели. В Triad принято графовое представление модели. Модель является иерархической, т.е. каждая вершина в слое структур может быть расшифрована подструктурой.

Входной язык описания моделей содержит переменные типа «модель». Над моделями определены операции. Операции определены как для моделей в целом, так и для каждого слоя (в Triad модель представлена слоем структур, рутин и сообщений.). Имитационная модель может быть описана средствами языка Triad или построена в результате исполнения некоторого алгоритма преобразования модели. Подсистема анализа модели обеспечивает получение информации по заранее сформулированному запросу, а не ограничивать пользователя строго регламентированным набором собираемых данных. Такой подход к сбору информации позволяет избежать избыточности собранной информации или того, что она окажется недостаточной.

Рассмотрим особенности представления имитационной модели в Triad.Net, особенное внимание уделим программному механизму, используемому для исследования моделей: информационным процедурам. Именно этот механизм авторы разработки используют для реализации удаленного доступа к модели и ряда компонентов системы безопасности.

Представление имитационной модели

Описание имитационной модели в Triad состоит из трех слоёв [Mikov, 1995]: слоя структур (*STR*), слоя рутин (*ROUT*) и слоя сообщений (*MES*). Таким образом, модель в системе Triad можно определить как *M* = {*STR*, *ROUT*, *MES*}.

Слой структур представляет собой совокупность объектов, взаимодействующих друг с другом посредством посылки сообщений. Каждый объект имеет полюса (входные P_{in} и выходные P_{out}), которые служат соответственно для приёма и передачи сообщений. Слой структур представляют графом. В качестве вершин графа следует рассматривать отдельные объекты. Дуги графа определяют связи между объектами. Объекты действуют по определённому алгоритму поведения, который описывают с помощью рутины (*rout*). Рутина представляет собой последовательность событий, планирующих друг друга. Выполнение события сопровождается изменением состояния объекта. Рутина так же, как и объект, имеет входные (Pr_{in} и выходные Pr_{out}) полюса. Входные полюса служат соответственно для приёма сообщений, выходные полюса – для их передачи. В множестве событий рутины выделено входное событие e_{in}. Все входные полюса рутины обрабатываются входным событием. Обработка выходных полюсов осуществляется остальными событиями рутины. Для передачи сообщения служит специальный оператор *out* (*out* <сообщение> *through* <имя полюса>). Совокупность рутин определяет слой рутин *ROUT*. Слой сообщений (*MES*) предназначен для описания сообщений сложной структуры. Система Triad реализована таким образом, что пользователь может описать только один слой. Так, если возникает необходимость в исследовании структурных особенностей модели, то можно описать в модели только слой структур.

Информационные процедуры

Для сбора, обработки и анализа имитационных моделей в системе Triad.Net существуют специальные объекты – информационные процедуры и условия моделирования. Информационные процедуры и условия моделирования реализуют алгоритм исследования.

Информационные процедуры ведут наблюдение только за теми элементами модели (событиями, переменными, входными и выходными полюсами), которые *указаны* пользователем. Если в какой-нибудь момент времени имитационного эксперимента пользователь решит, что следует установить наблюдение за другими элементами или выполнять иную обработку собираемой информации, он может сделать соответствующие указания, подключив к модели другой набор информационных процедур. Информационные процедуры являются единственным средством системы для одновременного доступа к элементам модели, принадлежащим разным объектам. Именно с помощью информационных процедур пользователь может осуществить взаимодействие (в том числе и удалённое) с объектами модели во время имитации. Условия моделирования анализируют результат работы информационных процедур и определяют, выполнены ли условия завершения моделирования. Система имитации Triad.Net

располагает языковыми средствами для описания алгоритмов работы информационных процедур. При изменении значения переменной, за которой ведётся наблюдение, при выполнении события, указанного пользователем, или после прихода (передачи) сообщения на входной полюс происходит подключение информационной процедуры к конкретному элементу модели (посредством параметров интерфейса) и данные обрабатываются по заданному в информационной процедуре алгоритму

Информационные процедуры и условия моделирования используют и для сбора информации о поведении модели, о ее характеристиках и в подсистеме защиты информации (внутренний уровень).

Реализация удаленного доступа

В СИМ Triad реализован удаленный доступ к имитационной модели [Mikov, 2007]. Удаленное взаимодействие с моделью осуществляется информационными процедурами. Разработан портал «Имитационное моделирование» и удаленный доступ можно осуществить через него. Программное обеспечение, реализующее СИМ Triad разделено на следующие модули: а) IMPortal – Интернет-портал, основанный на метаданных, который может использоваться отдельно от всего остального проекта как самостоятельное приложение; б) TriadCore – библиотека типов, содержащая описание базовых структур, используемых в имитационной модели; в) TriadEditor – графический пользовательский интерфейс для редактирования моделей (TriadEditor используется в модуле IMPortal для предоставления пользователям системы возможности совместной удаленной работы над моделями); г) TriadClient – приложение Windows, которое можно применять при однопользовательской работе с моделью или когда нет доступа к Интернет; TriadService-сервис Windows, используемый для выполнения моделей.

Подсистема защиты имитационной модели

Поскольку СИМ Triad предоставляет пользователям возможность удалённого доступа к модели, то возникает необходимость в защите СИМ от внешних атак злоумышленников.

Кроме удаленного режима работы СИМ Triad предоставляет сервисы для локальной работы с имитационной моделью. Однако компоненты модели распределены по отдельным компьютерам вычислительной системы и обмениваются сообщениями во время выполнения. Следовательно, злоумышленник может вмешаться в работу СИМ, заменив нужное сообщение, а результаты проведения имитационного эксперимента в этом случае станут недостоверными. Это может повлечь за собой принятие неверных решений. Для решения этой проблемы необходимо, чтобы система обнаружения вторжений отвечала не только за защиту компонента IMPortal, но и проводила анализ внутренних сообщений, которыми обмениваются компоненты системы Triad.NET.

Итак, ко всем перечисленным модулям системы Triad.NET добавляется еще один модуль TriadSecurity, который представляет собой систему обнаружения вторжений в объектно-ориентированную распределенную систему имитации Triad.NET.

При разработке подсистемы безопасности было принято решение выбрать мультиагентный подход. В защиту этого решения можно выдвинуть следующие аргументы: а) эффективная реализация задач, связанных с работой в компьютерных сетях (поиск, распределенная обработка информации); б) гибкая настройка системы, возможность в рамках одной системы реализовать разные подходы к решению задачи, поручив их реализацию разным агентам; в) масштабируемость за счёт возможности добавления или удаления агентов; г) увеличение функциональности за счет введения новых агентов; д) дополнительная степень защиты от злоумышленников за счет децентрализованного управления агентами.

В модуле TriadSecurity выделяют два уровня: а) внешний уровень модуля защиты системы Triad.NET (TriadInterSecurity); б) внутренний уровень (TriadIntraSecurity). *Внешний уровень* модуля защиты находится на сервере для публикации Web-приложений, взаимодействует с модулем IMPortal и предназначен для защиты системы от несанкционированного вмешательства через удаленный доступ.

Внутренний уровень находится на сервере БД и предназначен для защиты системы от попыток взлома изнутри, то есть от сообщений злоумышленника, направленных на разрушение таких компонентов системы, как TriadCore и TriadService.

Для взаимодействия этих уровней в рамках подсистемы защиты был выделен специальный агент: *агент взаимодействия уровней защиты*. Основной функцией этого агента является оповещение агентов внешнего уровня защиты о результатах работы агентов внутреннего уровня защиты. Любая система обнаружения вторжений при выявлении атаки на защищаемую систему должна сформировать некоторую ответную реакцию. В нашем случае при формировании ответной реакции необходимо учесть результаты работы агентов как внешнего, так и внутреннего уровней защиты. Особенно актуальным это становится при проведении распределенной атаки на систему Triad.NET, когда часть злоумышленников проводит атаку через удаленный доступ к системе, а другая часть злоумышленников, авторизовавшись в качестве легальных пользователей системы, пытается нанести ей вред изнутри. Было принято решение о том, что функция формирования ответной реакции подсистемы защиты будет возложена на внешний уровень защиты. Связано это, прежде всего, с тем, что вероятность проведения атак через удаленный доступ выше, чем вероятность проведения атак изнутри системы, так как проведение внутренней атаки требует преодоления внешнего уровня защиты.

Таким образом, легко выделяется следующая *вертикаль взаимодействия уровней подсистемы защиты*: администратор системы Triad.NET – внешний уровень защиты модуля TriadSecurity – внутренний уровень защиты модуля TriadSecurity .

Любую мультиагентную систему можно описать следующими множествами [Тарасов, 2002]: *MAS* = {*A*, *E*, *R*, *Org*, *Act*, *Com*, *Ev*}, где *A* – множество всех агентов, способных функционировать в некоторой среде *E*, находящихся в определённых отношениях *R*, и взаимодействующих друг с другом, формируя некоторую организацию *Org*, обладающих рядом индивидуальных и совместных действий *Act* (стратегий поведений и поступков), включая коммуникативные действия *Com*, и характеризуется (как и отдельные агенты) возможностями эволюции *Ev*. Каждый из агентов может быть охарактеризован следующими множествами <*S*, *Prog*, *Eff*, *Arch*, *P*, *A*, *G*, *E*>, где *E* – среда, в рамках которой функционирует агент; *S* – множество рецепторов агента, с помощью которых он получает информацию о состоянии окружающей его среды; *Eff* – множество эффекторов, с помощью которых агент воздействует на окружающую среду; *P* – информация, которую агент получает от своих рецепторов; *A* – действия, которые совершает над окружающей его средой агент при помощи своих эффекторов; *Prog* – алгоритм работы с информацией, поступающей агенту от рецепторов, направленный на формирование действий агента над окружающей средой (*Prog*: *P*→*A*); *G* – множество целей агента; *Arch* – архитектура агента.

Итак, целью мультиагентной системы защиты является определение следующих параметров: а) типа проводимой атаки; б) компонента системы, на который проводится атака; в) времени начала атаки; компьютера, с которого проводится атака; г) реакции на вторжение. Были выделены агенты следующих типов: а) агент-датчик (агент-сенсор); б) агент выявления атаки внешнего уровня защиты; в) агент реагирования; г) агент выявления атаки внутреннего уровня; д) агент взаимодействия с пользователем; агент взаимодействия уровней защиты. Рассмотрим более подробно функции каждого из них. Взаимодействие агентов подсистемы представлено на рис.1.

Агенты-датчики

К источникам внешних угроз относятся сетевой трафик и журналы регистрации. В подсистеме Triad.Net реализованы агенты-датчики, собирающие информацию из сетевого трафика, поскольку информация о любом удаленном воздействии, так или иначе, содержится в сетевом трафике, а журналы регистрации используются как вспомогательный источник. Опасаясь замедления реакции подсистемы защиты при сборе и обработке информации о системных журналах, авторы приняли решение не реализовывать агенты-датчики, собирающие информацию из сетевых журналов. Агент-датчик является реактивным, поскольку ему не нужна ни развитая модель внешнего мира, ни внутренняя память, ни мотивация, зато чрезвычайно важна быстрота его реакции [Тарасов, 2002]. Количество агентов-датчиков зависит от

масштабов вычислительной сети, точное их число определяет администратор системы Triad.NET. Архитектура агента-датчика является достаточно простой, в его работе можно выделить следующие этапы: рецепторы агента захватывают сетевой пакет, передают его исполнительному модулю агента (множество *P*); исполнительный модуль агента проводит разбор заголовка сетевого пакета по определенному алгоритму, о котором будет рассказано позднее (алгоритм *Prog*); эффекторы агента выкладывают результаты его работы на «доску объявлений» (множество *A*).

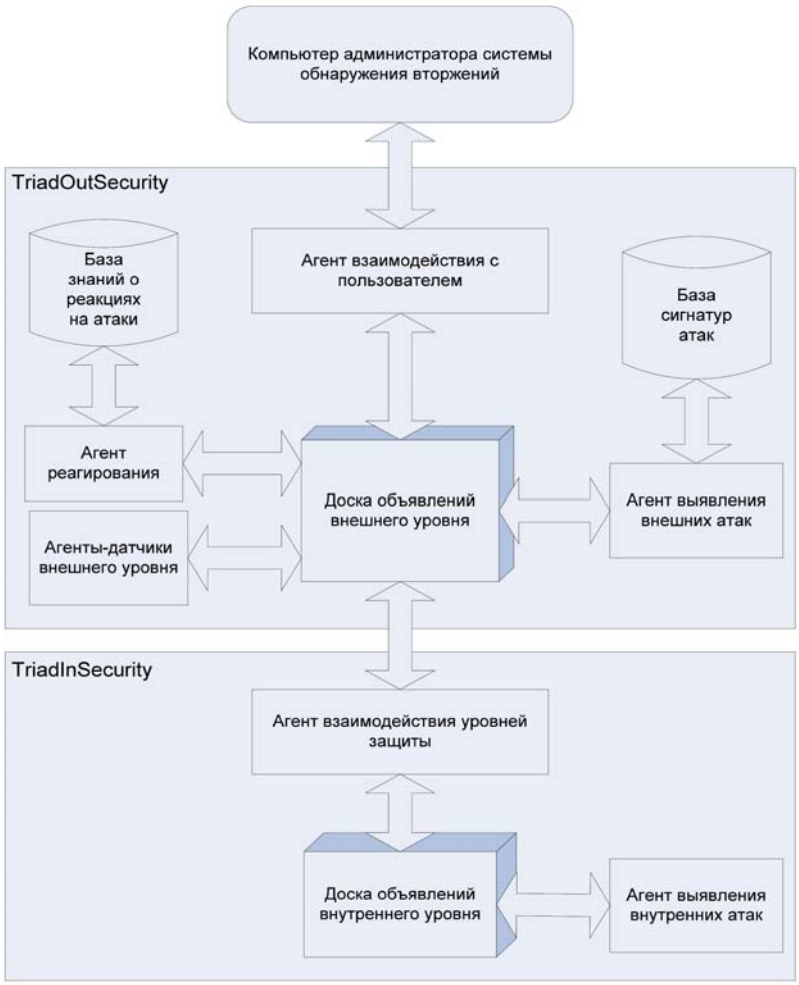

Рис.1. Взаимодействие агентов подсистемы защиты

Агент выявления атаки

Агент выявления атаки, несомненно, является самым важным компонентом всей подсистемы обнаружения вторжений, ведь именно ему предстоит решить самую сложную задачу – обнаружить попытку вторжения в систему Triad.NET (множество *G* целей агента). Этот агент реализуется как когнитивный агент с развитой моделью внешнего мира, у него должна быть своя внутренняя память, но главное – он должен уметь вести сложные и рефлексивные рассуждения. Основным компонентом агента выявления атаки является специальная база сигнатур атак. Под сигнатурой мы будем понимать некоторый признак трафика, определяющий одну из известных атак. База сигнатур состоит из правил. В базе сигнатур атак агент выявления атак хранит правила, описывающие различные виды атак, известные подсистеме обнаружения вторжений. Естественно, с появлением новых атак база сигнатур атак должна пополняться. Эту проблему должен решать сам агент выявления атаки, пополняя базу сигнатур. Кроме того, предусмотрен специальный компонент приобретения знаний, позволяющий администратору системы самому добавлять новые правила в базу сигнатур атак. Агент выявления атак функционирует

следующим образом: рецепторы агента сканируют «доску объявлений» с целью выявления новых записей с информацией о заголовке некоторого сетевого пакета и передают эту запись исполнительному модулю агента (множество *P*); исполнительный модуль агента, используя встроенный механизм вывода, базу сигнатур атак и внутреннюю (рабочую) память агента, анализирует эту запись на наличие информации о вторжении (алгоритм *Prog*); эффекторы агента выкладывают результаты анализа на «доску объявлений» (множество *A*).

Агент реагирования

Внешний уровень защиты формирует некоторую реакцию подсистемы обнаружения вторжений на выявленную атаку и предлагает этот вариант администратору системы. Формированием реакции внешнего уровня защиты занимается специально выделенный агент – агент реагирования (множество *G* целей агента). Это когнитивный агент, обладающий развитой моделью внешнего мира, внутренней памятью и мотивацией, кроме того, он должен иметь возможность проводить сложные рассуждения для определения верной реакции подсистемы защиты. Архитектура агента реагирования схожа с архитектурой агента выявления атаки за исключением того, что агент реагирования взаимодействует с базой знаний о реакциях на атаки. Функционирование агента реагирования можно описать следующим образом: рецепторы агента сканируют «доску объявлений» с целью выявления новых записей с информацией о подозрительных сетевых пакетах и передают эту запись исполнительному модулю агента (множество *P*); исполнительный модуль агента, используя встроенный механизм вывода, базу знаний о реакциях на атаки и внутреннюю (рабочую) память агента, анализирует эту запись с целью формирования ответной реакции подсистемы защиты (алгоритм *Prog*); эффекторы агента выкладывают результаты анализа на «доску объявлений» (множество *A*). При работе исполнительного модуля может сложиться ситуация, при которой агент реагирования не сможет по имеющимся у него данным сформировать реакцию системы. В этой ситуации агент реагирования обратится за помощью к своим рецепторам и потребует от них определенной информации с «доски объявлений» (например, он может потребовать информацию обо всех сетевых пакетах, пришедших с определенного IP-адреса).

Агент взаимодействия с пользователем

Назначение агента взаимодействия с пользоватерем – поддержка диалога с пользователем, которым в данной ситуации чаще всего будет администратор системы Triad.NET. Агент взаимодействия с пользователем является типичным реактивным агентом. Работа агента заключается в следующем: а) предоставление администратору информации о состоянии подсистемы защиты, а именно, списка сетевых пакетов, обрабатываемых подсистемой, с их параметрами (при этом администратор может динамически менять список отображаемых параметров); б) предоставление администратору информации об обнаруженных на систему Triad.NET атаках и возможные варианты реакции на создавшуюся ситуацию; в) предоставление администратору возможности запуска и остановки других агентов системы; г) предоставление администратору возможности различных настроек подсистемы безопасности.

Агент взаимодействия уровней защиты

Агент взаимодействия уровней защиты является самым простым агентом подсистемы защиты от вторжений, функцией этого агента является оповещение агентов внешнего уровня защиты о результатах работы агентов внутреннего уровня защиты. Безусловно, агент взаимодействия уровней защиты является реактивным, а его работа включает следующие действия: сканирование «доски объявлений» внутреннего уровня защиты в поисках любой новой информации и передача найденной записи эффекторам агента; добавление найденной записи на «доску объявлений» внешнего уровня защиты.

В предыдущих разделах данной работы были выделены агенты, входящие в состав мультиагентной подсистемы защиты TriadSecurity, были определены функции этих агентов и их архитектура. На следующем этапе необходимо спроектировать архитектуру уже всей подсистемы защиты, определив тем самым способ взаимодействия агентов.

Архитектура подсистемы защиты

В качестве базовой архитектуры для подсистемы защиты TriadSecurity была выбрана Blackboard-архитектура (архитектура «доски объявлений»), которая включает в себя следующие компоненты: «доска объявлений», которая отвечает за хранение сведений, доступных всем источникам знаний (агентам); источники знаний (агенты) – сущности, реализующие обработку этих сведений с целью решения тех задач, которые были поставлены как перед всей мультиагентной системой в целом, так и перед отдельно взятыми агентами этой системы. Результаты работы любого из агентов мультиагентной системы оказываются на «доске объявлений», после чего могут быть использованы любым другим агентом для достижения своих целей.

«Доска объявлений» – это тот компонент системы, который обеспечивает взаимодействие между всеми без исключения агентами подсистемы TriadSecurity, связывает всех агентов для решения общей задачи и является хранилищем всей информации о работе подсистемы обнаружения вторжений. Ранее упоминалось, что существует две «доски объявлений»: по одной на каждый уровень защиты. Вся информация с внутренней «доски объявлений» попадает и на внешнюю «доску объявлений».

С внутренней «доской объявлений» ведут работу всего два агента: агент выявления атак внутреннего уровня выкладывает на «доску объявлений» результаты своей работы, а агент взаимодействия уровней защиты сканирует «доску объявлений» с целью переноса всех новых записей на внешний уровень защиты. На «доске объявлений» внешнего уровня защиты выкладывается информация для множества агентов подсистемы защиты. Соответственно, возникает потребность в структуризации внешней «доски объявлений» с целью разграничения доступа к ней агентов подсистемы защиты: каждый агент подсистемы защиты сканирует информацию только на определенной части «доски объявлений», а не всю доску, что увеличивает быстродействие системы, а, следовательно, и быстроту ее реакции на вторжения.

Таким образом, «доска объявлений» внешнего уровня защиты представляет собой нормализованную базу данных, условно разделенную на три части (по той причине, что три агента взаимодействуют с «доской объявлений») с целью разграничения доступа агентов к «доске объявлений» и сокращения поиска агентами необходимой им информации.

Реализация подсистемы защиты имитационной модели

Архитектура приложения TriadSecurity базируется на принципах многослойной архитектуры, т.е. в рамках приложения выделяется несколько независимых слоев, каждый из которых работает со своими структурами данных и взаимодействует со смежными слоями, получая или, наоборот, отправляя запросы на обработку той или иной информации.

В приложении TriadSecurity выделяют следующие слои архитектуры: *Data Access Layer* (уровень доступа к данным), этот уровень отвечает за взаимодействие агентов с «доской объявлений» (класс DataAccessOperations реализует генерацию различных SQL-запросов от агентов подсистемы обнаружения вторжений к «доске объявлений»); *Agents Layer* (уровень работы агентов подсистемы), на этом уровне функционируют все агенты подсистемы обнаружения вторжений, за исключением агента взаимодействия с пользователем; *Presentation Layer* (уровень взаимодействия с пользователем), на этом уровне функционирует агент взаимодействия с пользователем, предоставляя администратору информацию о текущем состоянии системы.

Внутренний уровень подсистемы защиты имитационной модели

В настоящее время реализован внешний уровень подсистемы защиты имитационной модели. Внутренний уровень подсистемы защиты будет реализован с использованием механизма информационных процедур. Именно информационные процедуры являются удобным средством, которое может зафиксировать аномальное значение сообщения на входе объекта распределенной мимитационной модели, определить верную последовательность событий, выполняющуюся в модели, зарегистрировать неверное значение

переменной в заданный момент модельного времени и т.д. Кроме того, информационные процедуры являются единственным средством для доступа к различным объектам имитационной модели в конкретный момент времени.

Заключение

Итак, в статье рассмотрены вопросы реализации подсистемы защиты распределенной имитационной модели с удаленным доступом. В настоящее время наиболее полно реализован компонент, который отвечает за выявление внешних атак и защиту от них. В статье приведена архитектура подсистемы, при проектировании которой применен мультиагентный подход, рассмотрен перечень агентов, их назначение, их функции и аспекты реализации.

Благодарности

Статья частично финансированна из проекта **ITHEA XXI** Института Информационных теории и Приложений FOI ITHEA и Консорциума FOI Bulgaria (www.ithea.org, www.foibg.com).

Библиографический список

[Mikov, 1995] Mikov A.I. Simulation and Design of Hardware and Software with Triad // Proc.2nd Intl.Conf. on Electronic Hardware Description Languages, Las Vegas, USA, 1995. Pp. 15-20.

[Mikov, 2007] Alexander Mikov, Elena Zamyatina, Anton Firsov. Software for Remote Parallel Simulation// International Journal «Information Theories & Applications», Vol.14, № 4, Varna,2007, Pp. 389-395.

[Тарасов, 2002] Тарасов В.Б. От многоагентных систем к интеллектуальным организациям: философия, психология, практика. М.: Эдиториал УРСС, 2002. – 352 с.

Сведения об авторах

Александр Миков *– Кубанский государственный университет, профессор, заведующий кафедрой «Вычислительные технологии»; Россия, г. Краснодар, ул. Аксайская, 40/1-28;*
e-mail: alexander_mikov@mail.ru

Елена Замятина *– Пермский государственный университет, доцент кафедры математического обеспечения вычислительных систем, Россия, г. Пермь, 614017, ул. Тургенева, 33–40;*
e-mail: e_zamyatina@mail.ru

Михаил Панов *– Пермский государственный университет, выпускник кафедры математического обеспечения вычислительных систем, Россия, г. Пермь, ул. Юрша, 21-271;*
e-mail: panov-mikhail@yandex.ru