УДК 004.051 DOI: 10.17587/it.25.293-299

**Н. А. Грушо,** канд. физ.-мат. наук, e-mail: info@itake.ru,
**Е. Е. Тимонина,** д-р техн. наук, проф., e-mail: eltimon@yandex.ru,
Институт проблем информатики, Федеральный исследовательский центр "Информатика и управление" Российской академии наук, г. Москва

# Сравнение архитектур многоагентных систем[1]

*Эксплуатация новых или существующих информационных систем связана с необходимостью оперативного получения данных об инцидентах, возникающих в них, а также с оперативным выявлением причин этих инцидентов. Если информационные системы имеют множество узлов, а также присутствуют связи информационных систем между собой, то журналы событий, накапливаемые за небольшой промежуток времени, могут иметь огромное число записей. Естественно, что администраторы информационных систем не в состоянии за короткий промежуток времени проводить полный и подробный анализ таких журналов. В данной работе рассматриваются архитектуры многоагентных систем, осуществляющие сбор данных об инцидентах в информационных системах, в целях их дальнейшего автоматического анализа.*

***Ключевые слова:*** *агенты, многоагентные системы, архитектуры многоагентных систем, информационные технологии*

## Введение

Цифровая экономика подразумевает разработку и внедрение большого числа информационных систем (ИС), а также организацию связи между ними. Естественно, что по многим причинам могут возникать сбои в работе ИС. Сбои могут влиять на ИС локально, а могут приводить к глобальным сбоям, если речь идет о взаимосвязанных ИС. Для устранения таких сбоев необходимо оперативно определять источник сбоев. Это возможно, если имеется система, которая собирает данные с элементов ИС о запущенных процессах, действиях пользователей, сетевой активности, записях в журналах операционных систем и т. д. Активная работа пользователей или программного обеспечения, выполняющего основной функционал ИС, порождает большой объем данных, который необходимо оперативно доставлять для анализа в некое хранилище всех событий.

Для сбора данных о событиях, происходящих на элементах ИС, необходимо использовать специальное программное обеспечение (далее по тексту — Агенты). Агенты собирают данные о событиях, но сами не обрабатывают их. Обработка событий Агентами может привести к повышенной нагрузке на систему, в которой он работает. Поэтому Агенты должны передать собранные данные в единую базу данных событий (БДС). Только после этого отдельное программное обеспечение (SIEM) может провести анализ накопленных данных в БДС для получения результата — выявления причины сбоя [1]. Для построения такой системы сбора данных необходимо рассмотреть возможные простейшие способы построения многоагентных систем (МС), определить их слабые и сильные стороны, возможность их реализации.

Исследования в области разработки МС проводились многими исследователями [2], но в этих работах основное внимание акцентируется на способе и языке взаимодействия Агентов [3], интеллекте Агентов [4], групповом поведении многоагентной системы. Определение МС в этих работах примерно одинаковое [3] — это описание элементов этой системы и описание их взаимодействия между собой для достижения определенной цели. При этом чаще всего ставится вопрос о решении множества сложных задач: взаимодействие Агента с

[1]Работа выполнена при частичной поддержке РФФИ (грант 18-29-03081-мк).

окружающей средой и реакция на ее изменение, демонстрация определенного поведения Агентов для достижения цели, организация "социальных" отношений между Агентами [5]. В работах приводятся примеры МС для сложных инфраструктур, но нет описания способа их построения (архитектуры).

Одним из направлений исследований в области архитектур МС (АМС) является возможность ее реализации. В таких работах приводится классификация [6] или описание [7] готовой архитектуры без сравнения ее с другими возможными реализациями.

В найденных работах рассмотрены теоретические аспекты функционирования сложных интеллектуальных МС, а также приведены примеры готовых реализаций таких систем. В данной работе сравнение АМС начинается с рассмотрения простейшей архитектуры, определения ее преимуществ и недостатков, а также способов устранения выявленных недостатков с помощью дополнительных элементов и взаимосвязей, т. е. расширения архитектуры.

## Построение простейших архитектур многоагентных систем

Для построения АМС определим функции ее элементов и минимальные требования, предъявляемые к ним:

- сбор данных о действиях пользователей, программного обеспечения, сетевой активности, записях в журналах выполняет Агент, находящийся на наблюдаемом персональном компьютере, сервере (далее по тексту — ПК) или на выделенном персональном компьютере, сервере (далее — Внешний Агент). Агент способен накапливать события, но объем накопленных данных должен быть минимален, чтобы не повышать нагрузку на ПК. Необходимость наличия такого элемента рассмотрена в работах [8, 9];
- БДС находится на сервере, имеющем ограничение по числу входящих подключений, вычислительной мощности, но не имеющем ограничений по объему хранимых данных;
- линии связи между элементами АМС имеют одинаковую задержку и пропускную способность. Резервирование линий связи может существовать или отсутствовать.

После определения элементов и их функций можно создать первую, простейшую АМС.

На рис. 1 схематически изображена простейшая АМС. Агент, находящийся на ПК, проводит сбор данных и отправляет их в БДС. В случае если число подключаемых к БДС Агентов невелико, а линии связи гарантировано функционируют, то такая АМС является оптимальной. Но в случае если число Агентов превысит число максимально возможных подключений к БДС, а линии связи не будут гарантировано функционировать, то возникает ситуация двойного сбоя: БДС не способна принять подключения и данные от Агентов, а Агенты переполняют свое локальное хранилище накопленных данных. В таком случае АМС не сможет обеспечить гарантированный сбор, целостность данных и доставку их в БДС.

Для преодоления ограничения на число одновременных подключений необходимо ввести дополнительный, масштабируемый элемент в АМС, который бы обеспечил неограниченный рост числа подключаемых агентов. На рис. 2, 3 представлена АМС с таким элементом — Веб-сервером или Хранилищем. Этот элемент может быть представлен в виде кластеров серверов, он масштабируем и обеспечивает гарантированное принятие подключений от Агентов.

Важно учитывать, что введение в АМС промежуточных элементов приводит к увеличению стоимости ее реализации. Обслуживать такую АМС сложнее, так как серверы со специализированным программным обеспечением требуют наличия штата квалифицированных специалистов, запасных компонентов для оперативного ремонта и т. д.

Вернемся к рассмотрению ситуации, в которой линии связи могут отказать, что приведет к переполнению локального хранилища накопленных Агентом данных на ПК. Решений этой проблемы существует несколько:

- увеличение объема локального хранилища данных на ПК, на котором функционирует

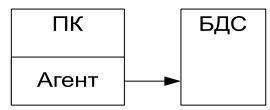


**Рис. 1. Простейшая АМС**


**Рис. 2. Дополнительный элемент АМС — Веб-сервер**

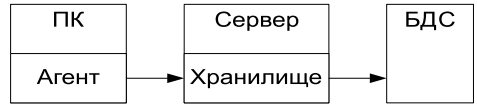


**Рис. 3. Дополнительный элемент АМС — Хранилище**

Агент. В случае же длительного отказа линий связи такое увеличение все равно может привести к переполнению;

- создание промежуточных локальных серверов для хранения накопленных данных (далее — Локальное Хранилище). Данный способ подразумевает, что линии связи в локальной сети не нарушены. Более того, это приводит к увеличению стоимости реализации всей АМС. Если же таких локальных сетей много, то увеличение стоимости будет значительным из-за большого числа дополнительных серверов — Локальных Хранилищ;
- выделение Агента на отдельный отказоустойчивый ПК (Внешний Агент) и осуществление удаленного сбора данных с остальных элементов АМС. Данный вариант приводит к значительному росту стоимости реализации АМС, увеличивает сложность обслуживания всей АМС, приводит к значительному усложнению программного кода Агента. Но он также имеет ряд преимуществ. В случае если ПК, на котором работает Агент, выходит из строя или подключение его к локальной сети нарушено, выявить неработающего Агента можно только по тому, что он перестает посылать данные в БДС. Если же Агент установлен на выделенном отказоустойчивом ПК, то он способен наблюдать не только за состоянием остальных ПК, но и может выявлять проблемы с сетевым оборудованием. Если линия связи Агента с БДС не функционирует, то такой Агент способен длительное время проводить сбор данных с элементов АМС в локальной сети.

Описанные выше варианты реализации АМС можно изобразить схематически (рис. 4, 5).

Если использовать все сильные стороны рассмотренных АМС, можно создать еще одну АМС, обеспечивающую избыточность получаемых от ИС данных (рис. 6).

В этой АМС должны присутствовать Агенты с увеличенным хранилищем накопленных данных, Внешние Агенты на отказоустойчивых серверах, Локальные Хранилища, промежуточные серверы — Хранилища или Веб-серверы, а также резервные линии связи между ключевыми элементами: Локальным Хранилищем — Хранилищем/Веб-сервером — БДС. Такая АМС в случае отказа отдельных элементов продолжит функционировать, позволяя проводить быстрое выявление причины возникших сбоев.

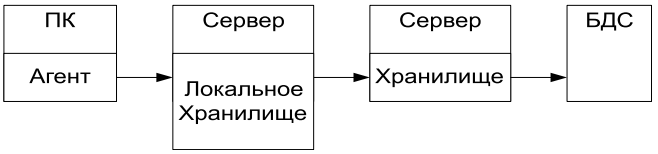


**Рис. 4. Локальное Хранилище в АМС**

Рассмотрим подробнее вопрос сравнения надежности архитектур. Оценка надежности архитектуры основана на оценке надежности каждого элемента. Например, существуют способы построения отказоустойчивых аппаратных платформ, но чаще всего к сбоям приводят не аппаратные платформы, а ошибки программного обеспечения (ПО) и нарушение в линиях связи между элементами. Рассмотрим отдельно каждый элемент архитектур.

Надежность работы Агента зависит от надежности ПО Агента, операционной системы (ОС), оборудования, на котором он работает. Выход из строя Агента, ОС, оборудования может быть определен исходя из того, что Агент перестает отправлять данные в БДС.

Надежность линии связи от элемента с Агентом до БДС, Веб-сервера, Хранилища или Локального хранилища зависит от сетевого оборудования в локальной сети. Если сеть не имеет специального отказоустойчивого оборудования, например, это простая офисная сеть, то обнаружить сбой в линии связи — это трудновыполнимая задача. Профессиональное сетевое оборудование имеет возможность диагностики сетевых портов и наличия подключения к узлу. В таком случае в сети должен

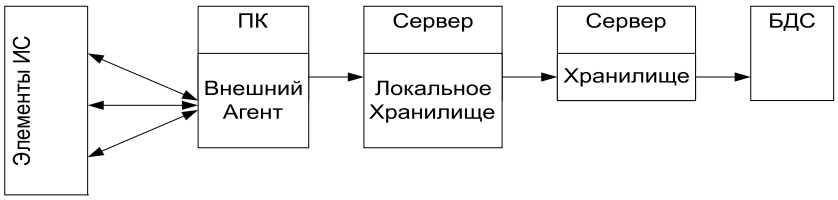


**Рис. 5. Внешний Агент в АМС**

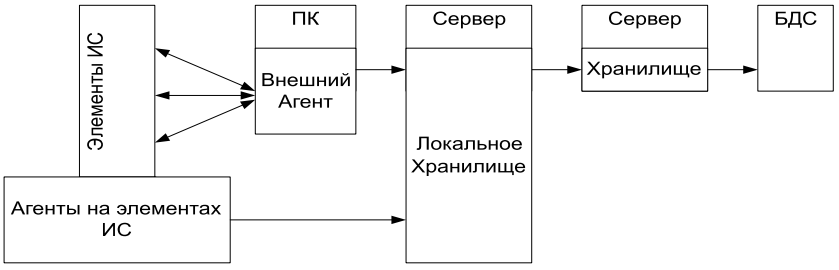


**Рис. 6. Избыточная АМС**

находиться Внешний Агент, наблюдающий за работой сетевого оборудования.

Надежность Веб-сервера, Хранилища, Локального Хранилища, БДС зависит от надежности аппаратной платформы и специального программного обеспечения, обеспечивающего предоставление сервисов. На данный момент существует множество решений, позволяющих организовать безотказную работу таких серверов. Благодаря резервированию и виртуализации можно создавать группы территориально удаленных серверов. В случае выхода из строя одной группы серверов или потери связи с ними Агенты могут использовать другую группу для передачи данных в БДС. Сами серверы БДС также могут быть выполнены в виде распределенной системы хранения данных, что позволит продолжить работу даже в случае отказа части серверов.

В итоге использование отказоустойчивых аппаратно-программных комплексов для реализации сервисов приема и обработки данных, резервирование линий связи и наличие Внешних Агентов, осуществляющих наблюдение за работой сетей, минимизация длины линии связи между Агентом и БДС позволяют реализовать любую из предложенных архитектур с необходимым уровнем надежности.

Теперь рассмотрим вопрос защищенности предложенных архитектур. Допустим, что одинаковые элементы в разных АМС имеют одинаковую степень защищенности. В таком случае отличия в степени защищенности АМС будут определяться степенью защищенности отличающихся элементов. Более подробно архитектурные уязвимости и способы их устранения рассмотрены в работе [10]. Например, сравнивая АМС на рис. 1 и АМС на рис. 2, можно сделать следующий вывод: АМС на рис. 2 имеет дополнительные элементы — это линия связи Веб-сервер — БДС и сам Веб-сервер. В таком случае АМС на рис. 2 имеет больше потенциальных уязвимостей и, следовательно, является менее защищенной. Такое же сравнение можно провести с другими рассмотренными архитектурами. Таким образом, можно сделать вывод, что увеличение числа элементов в АМС снижает защищенность, так как каждый новый элемент привносит в систему свои потенциальные уязвимости.

Сравнивая рассмотренные АМС между собой, важно учитывать время доставки данных от Агента в БДС. Допустим, что все линии связи между элементами архитектуры имеют одинаковые характеристики и обеспечивают минимальную задержку передачи данных, а серверы имеют достаточную производительность, чтобы на входе не образовывались очереди данных. В таком случае число соединений между Агентом и БДС определяет скорость доставки данных. Чем меньше линий связи, тем меньше задержка. В таком случае наилучший результат будет показывать архитектура "Агент — БД". Но если предположить, что БДС не может одновременно обрабатывать запросы от всех Агентов одновременно, то необходимо создавать промежуточные Хранилища, которые бы реализовывали принцип FIFO. В таком случае задержка определяется не только числом линий связи, но и длиной формируемой очереди на обработку данных. В случае, когда очередь разрастается, время задержки может быть существенным, однако это снижает нагрузку на БДС.

Таким образом, скорость доставки зависит не столько от АМС, сколько от вычислительной мощности БДС, Веб-сервера, Хранилища, а также от пропускной способности линий связи. При наличии достаточной вычислительной мощности можно реализовывать архитектуру "Агент — БД". Если же вычислительной мощности недостаточно или же линии связи не могут обеспечить необходимую пропускную способность, то можно использовать АМС типа "Агент — Хранилище — БДС". Аналогичная ситуация может возникать при использовании систем виртуализации для построения всей архитектуры ИС. Исследования влияния таких архитектур на скорость обмена данными между узлами приведены в работе [11].

## Определение параметров архитектур многоагентных систем

Каждая из рассмотренных АМС имеет свои преимущества и недостатки. Недостатки, выявленные у самой простой АМС, обусловливают необходимость добавить элементы, которые бы могли компенсировать их. Это приводит к усложнению АМС и, как следствие, к появлению новых недостатков. Следовательно, требуется добавление элементов, которые будут устранять уже их. Такое рассуждение приводит к созданию АМС из большого числа дорогостоящих элементов (рис. 6).

Перед разработчиком АМС возникает вопрос, какие параметры являются ключевыми при выборе той или иной АМС. Это может быть стоимость реализации АМС, ее надежность, защищенность, сложность обслужива-

ния, время передачи данных, время разработки Агентов и т. д.

Существует несколько способов определения параметров, влияющих на выбор АМС.

- *Аналитический.* Этот способ требует построения сложной модели, учитывающей все параметры АМС. Если же модель не будет учитывать каких-либо параметров (упрощенная модель), то выбор, сделанный с помощью нее, может оказаться не реализуемым на практике.
- *Экспериментальный.* Проведение экспериментов на макете АМС позволяет получить большое число результатов о поведении АМС в реальных условиях. В процессе создания макета АМС можно выявить множество ограничений, накладываемых реальностью: задержки и непредвиденные обрывы линий связи, искажение передаваемых данных, сбои и задержки в работе самих Агентов, перегрузки серверов и т. д.

Рассмотрим вопрос создания макета АМС, приведенной на рис. 1. Для создания такого макета требуется:

- ПК;
- программное обеспечение — Агент;
- подключение по локальной сети к серверу БДС;
- программное обеспечение — СУБД для организации БДС.

Реализация такого макета позволяет провести эксперименты, выявляющие нагрузку Агента на ПК, порождаемый им сетевой трафик, нагрузку на сервер БДС, а также дает возможность отработать ситуации с прерыванием связи. Для того чтобы этот макет реализовать в глобальной сети, потребуется установка сервера БДС на какомлибо хостинге. При этом сразу возникнет иная проблема — чаще всего хостинговые компании не предоставляют прямой доступ к СУБД из сети Интернет. Таким образом, макет по АМС на рис. 1 невозможно реализовать.

Рассмотрим возможность реализации макета АМС, рассмотренной на рис. 2. Для реализации этого макета требуется:

- ПК;
- программное обеспечение — Агент;
- подключение ПК Агента к сети Интернет;
- Веб-сервер с программным обеспечением, принимающим данные от Агентов;
- программное обеспечение — СУБД для организации БДС.

Чаще всего на хостингах используют язык программирования PHP и СУБД MySQL. Разработанный макет позволяет оценить АМС по следующим параметрам:

- параметры сбора данных Агентом;
- создаваемая Агентом нагрузка на ПК;
- создаваемая Агентом нагрузка на локальную сеть;
- создаваемая Агентом нагрузка на Веб-сервер;
- задержки, возникающие при передаче данных через сеть Интернет;

и т. д.

Для проведения оценки параметров АМС (см. рис. 2) был реализован макет. Агент разработан на языке C# и реализует следующий функционал:

- сбор данных о запущенных процессах с интервалом 10 мс. Данный интервал выбран исходя из того, что в системе иногда запускаются процессы, время работы которых мало. Для реальной системы отслеживания запускаемых процессов это время опроса может быть недостаточно малым;
- сбор данных о нажатиях на кнопки компьютерной мыши. Данный сбор реализован в виде перехвата системного вызова;
- сбор данных об установленных подключениях с интервалом 1 с. Данный параметр выбран исходя из того, что соединения не закрываются сразу после завершения сессии обмена данными, если речь идет о TCP/IP. Они переходят в состояние "ожидание завершения". Для отслеживания передачи данных с помощью UDP необходимо реализовать иной метод наблюдения, например, наблюдение за сетевыми пакетами;
- отправка данных на Веб-сервер каждые 10 с. Если обработка БДС с помощью специального программного обеспечения занимает незначительное время, то можно утверждать, что выявление сбоя на элементе системы будет обнаружено максимум через 10 с. В реальной системе это время может варьироваться в зависимости от предполагаемого времени обнаружения сбоя.

На Веб-сервере разработана программа на языке PHP, которая выполняет следующие функции:

- журналирование обращений Агента к Веб-серверу;
- прием данных от Агента и запись их в БДС.

Для проведения эксперимента Агент был установлен и запущен на ПК пользователя. Через 14 дней сбора данных от Агента были получены следующие результаты:

- среднее число запущенных процессов за сутки — 1235;
- среднее число нажатий клавиш — 4509;

- среднее число подключений — 13 197;
- среднее число нажатий на клавиши мыши — 6730;
- обращение агента за 1 сутки к БДС — 34 540;
- примерный объем БДС — 700 Мбайт.

Объем полученных данных, а также число обращений к БДС показывают, что сбор всех необходимых данных на макете приводит к большой нагрузке как на БДС, так и на сервер, принимающий данные от Агента (Веб-сервер). Следовательно, перед разработчиком такой системы встанет задача минимизации числа наблюдаемых параметров, а также уменьшения числа подключений для передачи данных. Например, для выявления отключения Агента от системы может быть достаточно реализовать "пульс", т. е. периодическое информирование БДС о том, что Агент функционирует. Для наблюдения за функционированием элемента ИС может быть достаточно наблюдения за одним процессом, выполняющим основной функционал ИС. Для предотвращения выхода из строя элемента необходимо наблюдать за параметрами оборудования, например, за состоянием жесткого диска. В таком случае потребуется информировать БДС только о событии ухудшения параметров, например, превышении допустимой температуры процессора или появлении ошибок на жестком диске. Эти решения приведут к снижению объема накопленных и передаваемых в БДС данных, а также позволят снизить число обращений к БДС.

Полученные в результате подобных экспериментов в реальных системах параметры могут быть использованы для создания прототипов рабочих АМС, оценок требуемой вычислительной мощности, требуемой пропускной способности линий связи, а также для проведения исследований по возможностям оптимизации.

## Заключение

В данной статье описаны способы построения простейших АМС для сбора данных. Также проведено сравнение АМС по различным критериям: число элементов, стоимость элементов, сложность создания и обслуживания, надежность архитектуры, защищенность, скорость доставки данных. В рассмотренных АМС выявлены как преимущества, так и недостатки, что привело к необходимости определения параметров, определяющих выбор той или иной АМС. Поскольку выбор АМС — это сложная задача, которая зависит от большого числа параметров, предложено использовать макетирование АМС. Цель макетирования — оценка параметров АМС в реальных условиях. Проведен эксперимент по макетированию АМС и сделан вывод о значимости получаемых экспериментальных данных.

Как и для любого проекта в области информационных технологий, реализация любой из АМС определена тремя базовыми параметрами: стоимость, качество, время. Уменьшение стоимости проекта вынуждает разработчиков использовать более дешевое оборудование и средства защиты информации, что приводит к снижению надежности и защищенности. Это снижает качество проекта по реализации АМС, но уменьшает время его реализации. Повышение качества приводит к увеличению стоимости и времени реализации, но при этом надежность, защищенность будет соответствовать необходимому уровню.

Оптимальное соотношение между временем реализации АМС, стоимостью реализации и качеством реализации АМС также может быть определено с помощью макетирования.

В данной статье не рассмотрены вопросы оптимизации сбора данных, так как не определен полный функционал системы сбора данных, но полученные в результате эксперимента данные показывают острую необходимость развития методов такой оптимизации.

### Список литературы


1. **Grusho A. A., Grusho N. A., Zabezhailo M. I., Timonina E. E.** Data mining in ensuring information security // Automatic Control and Computer Sciences, 2016. Vol. 50. No. 8. P. 722—725.

2. **Shehory Onn.** Architectural Properties of Multi-Agent Systems. Pennsylvania: Carnegie Mellon University, 1998. URL: https://pdfs.semanticscholar.org/c8f4/bd4f078fb2ac05650b23b9ec81ebf1613e57.pdf (дата обращения 11.02.19).

3. **Amit K. Chopra, Munindar P. Singh.** An Architecture for Multiagent Systems. An Approach Based on Commitments. URL: https://pdfs.semanticscholar.org/03bb/63477b3d30be1dde04a2a7071365e3cf87d5.pdf (дата обращения 11.02.19).

4. **Hyacinth S. Nwana.** Software agents: An overview // The Knowledge Engineering Review, 1996. Vol. 11, Iss. 3. P. 205—244.

5. **Jennings N. R., Sycara K., Wooldridge M. A.** Roadmap of Agent Research and Development // Autonomous Agents and Multi-Agent Systems, 1998. Vol. 1, Iss. 1. P. 7—38. URL: https://doi.org/10.1023/A:1010090405266 (дата обращения 11.02.19).

6. **Balaji P. G., Srinivasan D.** An Introduction to Multi-Agent Systems // Srinivasan D., Jain L. C. (eds) Innovations in Multi-Agent Systems and Applications — 1. Studies in Computational Intelligence, 2010. Vol. 310. Springer, Berlin, Heidelberg. P. 1—27.

7. **Reida M., Shakshuki E. M.** Implementing a Multi-agent System for Recording and Transmitting Biometric Information of Elderly Citizens // The 7th International Conference on Current and Future Trends of Information and Communication Technologies in Healthcare (ICTH 2017), 2017. P. 334—343.

8. **Грушо А. А., Забежайло М. И., Зацаринный А. А., Николаев А. В., Писковский В. О., Сенчило В. В., Судариков И. В., Тимонина Е. Е.** Об анализе ошибочных состояний в распределенных вычислительных системах // Системы и средства информатики. 2018. Т. 28, № 1. С. 99—109.

9. **Грушо А. А., Грушо Н. А., Тимонина Е. Е., Шоргин С. Я.** Возможности построения безопасной архитектуры для динамически изменяющейся информационной системы // Системы и средства информатики. 2015. Т. 25, № 3. С. 78—93.

10. **Грушо А. А., Грушо Н. А., Тимонина Е. Е., Шоргин С. Я.** Архитектурные уязвимости распределенных информационно-вычислительных систем. Системы и средства информатики. 2016. Т. 26, № 3. С. 74—82.

11. **Грушо А. А., Грушо Н. А., Левыкин М. В., Тимонина Е. Е.** Безопасные архитектуры распределенных информационно-вычислительных систем на основе комплексной виртуализации // Проблемы информационной безопасности. Компьютерные системы, 2016. Вып. 4. С. 31—35.

**N. A. Grusho,** Candidate of Science (PhD) in physics and mathematics,
e-mail: info@itake.ru
**E. E. Timonina,** Doctor of Science (Tech), Professor, e-mail: eltimon@yandex.ru
Institute of Informatics Problems, Federal Research Center "Computer Sciences and Control"
of the Russian Academy of Sciences, Moscow

# Comparison of Architectures of Multi-Agent Systems

*Usage of the new or existing information systems is connected with need transfer of data acquisition about the incidents arising in them and also with transfer identification of the reasons of these incidents. If information systems have a set of hosts and also there are communications of information among them, then the logs accumulated for a small period can have a huge amount of records. It is natural that administrators of information systems are not able to carry out the full and detailed analysis of such information for a short period. In this paper architectures of multi-agent systems which are carrying out collection of data on incidents in information systems for the purpose of their further automatic analysis are considered.*

***Keywords****: agents, multi-agent systems, architecture of multi-agent systems, information technologies*

DOI: 10.17587/it.25.293-299

**Acknowledgements:** The work was carried out with the partial support of the Russian Foundation for Basic Research (grant 18-29-03081-microns).

**References**

1. **Grusho A. A., Grusho N. A., Zabezhailo M. I., Timonina E. E.** Data mining in ensuring information security, *Automatic Control and Computer Sciences*, 2016, vol. 50, no. 8, pp. 722—725.

2. **Shehory Onn.** Architectural Properties of Multi-Agent Systems. Pennsylvania: Carnegie Mellon University, 1998, available at: https://pdfs.semanticscholar.org/c8f4/bd4f078fb2ac05650b23b9ec81ebf1613e57.pdf (date of access: 11.02.19).

3. **Amit K. Chopra, Munindar P. Singh.** An Architecture for Multiagent Systems. An Approach Based on Commitment, available at: https://pdfs.semanticscholar.org/03bb/63477b3d30be1dde04a2a7071365e3cf87d5.pdf (date of access:11.02.19).

4. **Hyacinth S. Nwana.** Software agents: An overview, *The Knowledge Engineering Review,* 1996, vol. 11, iss. 3, pp. 205—244.

5. **Jennings N. R., Sycara K., Wooldridge M. A.** Roadmap of Agent Research and Development, *Autonomous Agents and Multi-Agent Systems*, 1998, vol. 1, iss. 1. P. 7—38, available at: https://doi.org/10.1023/A:1010090405266 (date of access: 11.02.19).

6. **Balaji P. G., Srinivasan D.** An Introduction to Multi-Agent. Innovations in Multi-Agent Systems and Applications — 1. *Studies in Computational Intelligence*, 2010, vol. 310. Springer, Berlin, Heidelberg, pp. 1—27.

7. **Reida M., Shakshuki E. M.** Implementing a Multi-agent System for Recording and Transmitting Biometric Information of Elderly Citizens, *The 7th International Conference on Current and Future Trends of Information and Communication Technologies in Healthcare (ICTH 2017*), 2017. P. 334—343.

8. **Grusho A. A., Zabezhailo M. I., Zatsarinnyy A. A., Nikolaev A. V., Piskovski V. O., Senchilo V. V., Sudarikov I. V., Timonina E. E.** *Ob analize oshibochnykh sostoyaniy v raspredelennykh vychislitel'nykh sistemakh* (About the analysis of erratic statuses in the distributed computing systems), *Sistemy i Sredstva Informatiki*, 2018, vol. 28, no. 1, pp. 99—109 (in Russian).

9. **Grusho A., Grusho N., Timonina E., Shorgin S.** *Vozmozhnosti postroeniya bezopasnoy arkhitektury dlya dinamicheski izmenyayushcheysya informatsionnoy sistemy* (Possibilities of Secure Architecture Creation for Dynamically Changing Information Systems), *Sistemy i Sredstva Informatiki*, 2015, vol. 25, no. 3, pp. 78—93 (in Russian).

10. **Grusho A. A., Grusho N. A., Timonina E. E., Shorgin S. Ya.** *Arkhitekturnye uyazvimosti raspredelennykh informatsionno vychislitel'nykh sistem* (Architectural vulnerabilities of the distributed information systems, *Sistemy i Sredstva Informatiki*, 2016, vol. 26, no. 3, pp. 74—82 (in Russian).

11. **Grusho A. A., Grusho N. A., Levykin M. V., Timonina E. E.** *Bezopasnye arkhitektury raspredelennykh informatsionno-vychislitel'nykh sistem na osnove kompleksnoy virtualizatsii* (Secure architecture of distributed information systems on the basis of integrated virtualization), *Problemy Informatsionnoy Bezopasnosti. Komp'yuternye Sistemy*, 2016, no. 4, pp. 31—35 (in Russian).