УДК 681.3.06

А.В. Боярчук, В.С. Харченко

Национальный аэрокосмический университет им. Н.Е. Жуковского «ХАИ», Харьков

АНАЛИЗ СОВРЕМЕННЫХ РАЗРАБОТОК И ТЕХНОЛОГИЙ В ОБЛАСТИ ПОСТРОЕНИЯ ГАРАНТОСПОСОБНЫХ WEB-СЕРВИСОВ

Рассмотрены технологии разработки отказоустойчивых web-служб и приложений. Проанализированы методы планирования производительных систем на основе web-сервисов. Осуществлен анализ обеспечения безопасности web-сервисов.

планирование производительных систем, безопасность, гарантоспособность, web-сервис

Введение

Постановка задачи. Архитектура web-сервисов в настоящее время является одной из наиболее быстро развивающихся ИТ-сфер, поскольку уже стала фактически стандартом для взаимодействия различных приложений, функционирующих на гетерогенных платформах. Эта архитектура поддерживает разработку и внедрение открытых систем, в которых разработка и интеграция различных компонентов может выполняться как на этапе разработки системы в целом, так и во время ее функционирования. Индивидуальные компоненты (web-сервисы) представляют свои услуги через реестр (обычно разработанный с использованием UDDI-стандарта). Услуги описываются XML-подобным языком, который носит название WSDL (Web Services Definition Language).

Web-сервисы в настоящее время широко используются в разработке многочисленных критических приложений, среди которых следует особо отметить Интернет-банкинг, онлайн-магазины, системы резервирования и продажи туристических услуг, системы электронного бизнеса и электронной науки. Именно поэтому исследования по повышению надежности подобного рода систем являются перспективным направлением развития передовых информационных технологий.

На основании проведенного анализа работ [1, 19] можно выделить следующие ключевые аспекты концепции гарантоспособности (dependability), или надежности в широком смысле слова [2]:

– гарантоспособность – это способность компьютерной системы гарантированно предоставлять набор услуг (сервисов). При этом понятие отказа формулируется исходя из корректности и полноты сервисов, предоставляемых пользователю;

– в свою очередь, гарантоспособность включает свойства готовности (availability, готовности предоставления корректного сервиса), безотказности (reliability, продолжительности корректного сервиса), безопасности (safety, отсутствия катастрофических последствий для пользователей и окружающей среды), конфиденциальности (confidentiality, отсутствия неавторизированного доступа к информации), целостности (integrity, отсутствия недопустимого или непредусмотренного изменения состояния системы) и обслуживаемости (maintainability, приспособленности к ремонту и модификации), что продемонстрировано на рис. 1;

– отказоустойчивость (fault-tolerance) определяется как средство поддержания гарантоспособности и его составляющих.

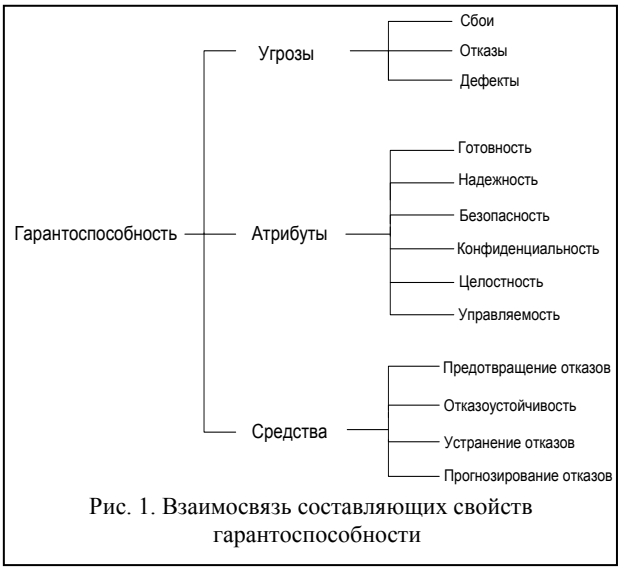

Рис. 1. Взаимосвязь составляющих свойств гарантоспособности

В современных исследованиях рассматриваются вопросы, которые представляют новые технологии и механизмы по разработке отказоустойчивых приложений сервис-ориентированной архитектуры, построению производительных систем web-сервисов, а также обеспечению безопасности web-комплексов.

Существует три подхода к обеспечению и оценке гарантоспособности приложений сервис-ориентированной архитектуры: «атрибутивный» подход подразумевает анализ составляющих свойств гарантоспособности, «архитектурный» компонент изучает различные варианты архитектуры систем web-сервисов; «технологический» аспект фокусируется на технологиях, применяемых для разработки и функционирования web-сервисов (UDDI, WSDL, SOAP).

Цель статьи – проведение компаративный ана-

лиз современных разработок и технологий в области построения гарантоспособных web-сервисов. Проведенный анализ не претендует на полноту, поскольку эта проблема в целом очень объемна и не может быть решена путем изучения некоторого количества трудов. В ограниченных рамках статьи ставится задача выделить ключевые задачи, поставленные разработчиками гарантоспособных приложений, и ознакомиться с достигнутыми результатами с учетом применения последних разработок в области web-инжиниринга.

1. Разработка отказоустойчивых служб и приложений сервис-ориентированной архитектуры

В статьях [1, 3] представлен инновационный подход для улучшения готовности и надежности web-сервисов. Смысл представленного подхода состоит в вызове «контейнеров», устойчивых к отказам. С помощью этих обращений к контейнерам вызываются внешние службы, к которым таким образом добавлена устойчивость к отказам. Это достигается благодаря тому, что контейнеры могут быть сконфигурированы с помощью определенной политики, которая определяет, какой из механизмов обеспечения отказоустойчивости будет применен к службам данного контейнера. Контейнер вызывает свои службы, пропуская их через реплики в шаблоне, установленном с помощью определенной политики. Приведенное инструментальное средство и SDK упрощают создание и развертывание контейнера и его политик.

Разработанная архитектура предлагает стойкий механизм для достижения определенных типов отказоустойчивости. Приведенная методика требует минимальных затрат усилий и времени. Избыточность достигается на уровне служб и, следовательно, на уровне программного и/или аппаратного обеспечения. Более того, расширяемость данной архитектуры предоставляет пользователю возможность модификации и дополнения множества механизмов обеспечения отказоустойчивости по сравнению с первоначально имеющимся объемом.

Данная архитектура имеет следующие преимущества. Во-первых, корневые рабочие станции с репликами могут быть различны, реализация служб на каждой машине тоже может быть различна. Во-вторых, описанный подход может быть с успехом применен и для сторонних (внешних) служб. Поиск и подключение реплицированных служб на этапе времени выполнения возможен благодаря механизму позднего связывания. Наконец, избыточность служб может быть обеспечена без чрезмерных затрат финансовых ресурсов (по сравнению с реплицированием серверов).

Иерархическая структура моделирования для оценки надежности Интернет-приложений, приведенная в статье [4], проиллюстрирована на примере туристического агентства. Для моделирования системы используются четыре уровня: пользовательский, функциональный, уровни служб и ресурсов. Первый уровень описывает, как пользователи работают с приложением, три остальных уровня демонстрируют, каким образом пользовательские запросы обслуживаются приложением на разных уровнях абстракции. Воспринимаемая пользователем степень готовности базируется на влиянии отказов, связанных как со спецификой функционирования данной системы, так и способами стандартного поведения аппаратных и программных средств.

Для выполнения своих функций веб-система турагенства взаимодействует с открытым интерфесом системы бронирования авиабилетов (AirFrance, KLM,…), системами бронирования отелей (Sofitel, Holiday Inc, …) и сервисами бронирования автомобилей (Hertz). Веб-система турагенства функционально представлена в виде двух базовых компонентов: клиентского и серверного. Клиентская часть управляет потоками пользовательского ввода, осуществляет необходимые проверки и перенаправляет данные на сторону серверного компонента. Последний является главным функциональным модулем системы. Он выполняет все необходимые операции по всем входящим клиентским запросам – проверяет доступность запрашиваемых элементов заказа (билет, отель, автомобиль), бронирует и оплачивает выбранные услуги. Это реализуется с помощью транзакций с системами резервирования соответствующих услуг, конвертирования поступающих запросов к форматам, воспринимаемым целевыми сервисами, и управления возникающими исключениями.

Идея оценки готовности состоит в комбинировании результатов, полученных от двух моделей: модели представления и модели готовности. Модель представления воспринимает приход запросов и служебные процессы, оценивая параметры системы, определенных в модели готовности. Модель готовности используется для оценивания вероятности устойчивости системы на основании возникновения отказов и восстановления системы после них. Этот подход основывается на том предположении, что система находится в квазиустойчивом состоянии между возникновением циклов «отказ-восстановление». Это допущение является верным в том случае, когда промежуток времени между отказами и восстановлениями системы велик по сравнению с интервалом времени между заявками.

Предложенная иерархическая модель предлагает системный и прагматичный подход к оценке характеристик надежности целевого приложения на различных уровнях абстракции.

В работе [5] отмечен тот факт, что большинство существующих на сегодняшний день схем обеспечения отказоустойчивости веб-серверов обнаруживают отказ сервера и перенаправляют поток клиентских запросов на запасной сервер. Эти технологии в большинстве своем не обеспечивают прозрачного

управления запросами, находящимися на обслуживании в момент возникновения отказа сервера. Разработан прозрачный для клиента механизм обеспечения отказоустойчивости веб-серверов, который гарантирует корректное управление обрабатываемыми запросами в момент возникновения отказа сервера. Эта схема основывается на запасном сервера, находящемся в режиме ожидания и простых прокси-серверах. Механизмы управления ошибками, заложенные в протокол TCP, применяются для транслирования запросов на первичный и запасной сервера, а также для достоверной доставки ответа от сервера, на котором может произойти сбой в момент отсылки ответа. Предлагаемая схема не предполагает изменений на уровне ядра ОС или использования реализаций TCP на пользовательском уровне и требует лишь внесения минимальных изменений в структуру серверного программного обеспечения.

Схемы обеспечения отказоустойчивости могут быть с успехом использованы для повышения готовности и надежности сетевых сервисов [6]. Одним из аспектов функционирования этих схем – реконфигурация (service failover) доступных ресурсов и восстановление состояния, необходимого для продолжения функционирования системы, несмотря на потерю части ресурсов и повреждение данных. Ранее была презентована схема CoRAL (Connection Replication and Application-level Logging), методология обеспечения отказоустойчивости веб-сервисов, основанная на резервированном вторичном сервере и протоколировании состояний системы. В случае возникновения отказа первичного сервера, активные клиентские соединения «откатываются» до того состояния, пока они обрабатывались непрерывно и в штатном режиме. Если дополнительные ресурсы сервера в данный момент доступны, новый сервер может быть ре-интегрирован в систему для возвращения ее к отказоустойчивой операции. По результатам проведенных тестов данной схемы можно говорить о коротких временных интервалах отказов системы и о незначительной перегрузке системы в течение периода между отказами.

Отдельный интерес представляет гипотеза [7], утверждающая, что такие технологии восстановления системы после отказов, как дублирование процессов, могут парировать большинство отказов программного обеспечения без модификация исходного кода приложения или использования специальных программных «надстроек». Авторы идеи проанализировали. В результате анализа отказов, возникающих в трех наиболее распространенных программных продуктах с открытым кодом: веб-сервере Apache, оконном менеджере GNOME и СУБД MySQL, была выявлена следующая закономерность: от 72% до 87% отказов не зависят от среды выполнения и, следовательно, являются детерминированными. Восстановление после этих отказов требует применения определенных знаний, специфичных для каждого программного продукта. Половина оставшихся отказов зависит от условий среды выполнения, и восстановление после них также требует специфичных сведений о приложениях. К сожалению, только 5 – 14% были обнаружены и зафиксированы в момент переходных условий, таких как синхронизация. Классические технологии восстановления приложений после отказов (дублирование процессов), не являются достаточными для восстановления после всех типов отказов.

Большинство технологий, используемых в настоящее время для повышения готовности веб-серверов, не предлагают решений, которые бы обеспечивали устойчивость от отказов для запросов, находящихся в стадии обработки в момент возникновения отказа. Некоторые другие схемы требуют наличия серверов с детерминированным поведением в любой ситуации или внесения изменений в клиентское программное обеспечение (веб-браузер). Основой реализации схемы обеспечения отказоустойчивости [8] является сервер в режиме горячей замены, управляющий журналами поступления запросов и отправки ответов. Эта схема требует модификации ядра ОС Linux и веб-сервера Apache с использованием соответствующих механизмов взаимодействия их модулей.

Другой путь обеспечения отказоустойчивости состоит в применении решения [9], основанном на упреждающем восстановлении системы после отказа, основанием для которого служит применение гарантоспособных композитных (интегрированных) веб-сервисов. Такой механизм поддержки транзакций на уровне каждой службы не влияет на автономность индивидуальных веб-сервисов. Его суть состоит в структуризации системы с помощью кооперативных атомарных операциях, поведение которых строго регламентировано как в случае штатного функционирования системы, так и в случае ее отказа. Для формализации этого процесса определена нотация Web Service Composite Actions (WSCA), базирующаяся на основе концепции координированных атомарных операций, которая позволяет структурировать композитные веб-сервисы в терминах гарантоспособных операций. Таким образом, отказоустойчивость может быть достигнута как результат агрегирования нескольких потенциально ненадежных веб-сервисов.

Представленная теоретическая конструкция получает свою практическую реализацию благодаря стратегии разработке композитных веб-сервисов на основе WSCA-технологии, которая подразумевает использование спецификации WSCA на языке XML.

Главной проблемой для разработчиков комплексных веб-приложений является распределение и согласование функций, выполняемых различными модулями системы, с одной стороны, и настройка ком-

понентов для парирования многочисленных ошибок с другой стороны.

Управление исключениями в таких системах является далеко не тривиальной задачей. Для решения этой задачи можно используют механизм т.н. «координированных атомарных операций» [10] для обработки исключительных ситуаций путем рекурсивной структуризации интегрированной системы и асоциирования обработчиков исключений с такими операциями. Для имплементации такого подхода разработан специальный механизм защищенных «конвертов»: каждая операция любого компонента преобразуется в атомарную операцию с хорошо определенным интерфейсом. Для улучшения комбинированного использования нескольких сред окружения, формируемых на этапе выполнения (runtime packages), разработчики внедрили механизм многоуровневой обработки исключений.

2. Разработка производительных систем

Проблема построения производительных и надежных систем сегодня представлена оригинальными разработками, среди которых можно отметить следующие.

Так, в статье [11] представлены две схемы построения оптимизированной веб-системы, которая уменьшает нагрузку на сервер. Для динамического контента пропускная способность серверного кластера значительно может быть увеличена за счет разделения задач по генерации контента между первичным и вторичным контентом. Для статического контента, который не требует значительны серверных услуг по генерированию, загрузка сервера может снизиться вследствие отключения функции подробного протоколирования всех событий. Реализация проекта, представленного разработчиками – авторами статьи, подразумевает внесение изменение в ядро ОС Linux и веб-сервер Apache. В статье рассматриваются технические детали реализации и анализируются показатели загрузки сервера.

При использовании компоненто-ориентированной технологии разработки программного обеспечения (CBSD) возможные преимущества разработки COTS ПО (небольшая стоимость и риски, высокое качество) не могут быть достигнуты в достаточной степени вследствие неадекватного или неполного предложения для спецификации компонентов. В частности, нефункциональные аспекты играют большую роль для описания и выбора компонентов, но, в то же время, не используются для документирования и поиска компонентов. Концепция [12] предлагает такой вариант документирования COTS-компонентов, при котором нефункциональные аспекты компонентов могут быть описаны, и трейдеры имеют возможность эти описания для эффективного поиска и выбора соответствующих компонентов. Такой подход предполагает естественную интеграцию с RE-структурой, декомпозицию запросов и трассируемость.

Суть методологии [13] состоит в том, чтобы пользователь сам имел возможность задавать свои предпочтения при работе с информационной системой (пользовательские настройки, допустимые задержки, ограничения и другие измеримые характеристики поведения системы), после чего система автоматически проводила свою реконфигурацию в соответствии с поступившими требованиями. В основу предлагаемого grid-сценария положен тот факт, что большое количество часто используемых сервисов будет предложено различными провайдерами, что позволит удовлетворить каждый индивидуальный запрос любым множеством реализаций служб, идеально диверсифицированных и независимых по целому спектру параметров: готовности, стоимости, точности и т.д. Некоторые из этих реализаций смогут удовлетворить запросы лучше, чем другие; использование же нескольких из них вместе (как последовательные серии, в режиме «телеметрии», или в другой конфигурации) поможет достичь требуемого результата при оптимальных параметрах.

Три взаимосвязанные функциональные компоненты информационной системы – безопасность, отказоустойчивость и оценки качества программного продукта – является темой статьи можно рассматриваться как различные фасеты гарантоспособности [14]. Диверсность может применяться на различных уровнях функционирования системы: поддержка выполнения задач (аппаратное обеспечение плюс операционная система), условия выполнения и проектирование программных средств, интерфейс взаимодействия пользователя и программных средств. Кроме этого, диверсность может применяться для валидации dependable-систем в течение всего периода их проектирования и разработки.

Работа [15] описывает тренды развития технологий веб-сервисов, которые поддерживают технологию динамического взаимодействия (Dynamic Collaboration). Для реализации этой технологии и ее применения проиллюстрированы пять основных ее характеристик: взаимодействие и интероперабельность, безопасность и защищенность, робастность и надежность, динамичность и связанность. Для определения этих характеристик разработаны и применяются многочисленные спецификации на XML.

Как известно, одним из самых важных нефункциональных требований к SQL-серверам является их производительность и надежность. В результате анализа [16] функционирования диверсных SQL-серверов можно сделать вывод о диверсном дизайне как способе разработке надежных системы управления базами данных. Приводится обзор фундаментальных решений и допущений, в соответствии с которыми функционируют приложения для реплицированных СУБД. В качестве замечания по сути работы СУБД принят тот

факт, что СУБД может работать или полностью корректно, или не работать вовсе. Этот обзор основывается на анализе ошибок 4 SQL-серверов, который подтверждает, что большое количество отказов СУБД не являются критичными, т.е. не приводят к аварийному завершению работы системы и потере данных. Анализируются различные варианты разработки решения для репликации данных – от размещения нескольких реплик на одном SQL-сервере до размещения реплик на различных диверсных SQL-серверах. В итоге можно сделать вывод о том, что использование SQL-серверов в некоторых случаях улучшает время отклика системы на запрос пользователя.

3. Обеспечения безопасного функционирования web-сервисов

Избыточность и диверсность – самые широко распространенные принципы построения отказоустойчивых систем, способных парировать любые отказы. Применение избыточности без диверсности может привести к тому, что система окажется неспособной отразить систематическую атаку. Так, в статье [17] анализируются роли, которые отводятся избыточности и диверсности в части построения отказоустойчивых и защищенных систем, отдельное внимание уделено сфере использования этих технологий для обнаружения внешних вторжений в систему. Исследуются различные способы применения избыточности и диверсности для обеспечения безопасности. На основании построения и анализа вероятностной модели функционирования системы можно отметить факт возможности применения свойств избыточности и диверсности для построения систем не только с высокой готовностью и защищенностью (safety), но и безопасностью (security). При этом следует отдельное внимание уделять факторам, обеспечивающим высокую эффективность применения избыточности и диверсности, и роли «независимости» между уровнями защиты.

Обеспечение безопасности веб-сервисов на уровне приложений зачастую приводит к возникновению уязвимостей, присущих самому коду веб-приложения (вне зависимости от технологий, с помощью которых было разработано данное приложение или СУБД, с которым работает приложение) – к такому выводу пришли авторы исследования [18]. За последние несколько месяцев значительно возросло количество несанкционированных вторжений на уровне приложения: покупка товаров в онлайн-магазинах без оплаты тарифа по доставке товара, хищение пользовательских имен, паролей и номеров кредитных карт.

Предлагаются новые инструментальные средства и технологии, частично решающие проблему обеспечения безопасности веб-приложений. Подход заключается в создании масштабируемого механизма, который абстрагирует политику безопасности от большого количества веб-приложений, разработанных в различных гетерогенных мультиплатформенных средах разработки.

Последней – по порядку, но не по значению – работой, рассматривающей вопросы безопасности web-систем, представлена статья [19]. Исходная версия консорциума W3C протокола SOAP не содержит свойств и методов для поддержки безопасности, и представляет спецификацию SAML (Security Assertion Markup Language – Язык разметки утверждений безопасности), определяющую перечень схем, ориентированных на поддержку безопасности, для структуризации документов. SAML выступает в качестве рамочной структуры для обмена данными аутентификации и утверждениями безопасности между многочисленными участниками информационного взаимодействия в Интернет, реализуемого с использованием протоков HTPP и SOAP.

Заключение. Направления дальнейших исследований

Таким образом, в данной работе был проведен анализ концепции гарантоспособности и приведен обзор современных исследований в области построения надежных web-сервисов, отказоустойчивых web-приложений, а также в сфере обеспечения безопасности web-сервисов. В результате можно отметить тот факт, что специфика архитектуры web-сервисов по сравнению с «традиционными web-приложениями» накладывает свой отпечаток на работу по обеспечению отказоустойчивости, производительности и безопасности приложений сервис-ориентированной архитектуры [20]. Вследствие сравнительной новизны самой концепции web-сервисов в настоящее время остаются открытыми вопросы, связанные с уязвимостями web-сервисов и способами их защиты [21].

На основании проведенного компаративного анализа упомянутых исследований, посвященных проблеме технологий разработки отказоустойчивых Web-сервисов, можно с уверенностью говорить о WSCA-технологии как об одном из наиболее мощных и в то же время гибких инструментов для проектирования отказоустойчивых Web-сервисов из ненадежных компонентов. В качестве аргументов, подтверждающих этот факт, можно привести такие достоинства WSCA-технологии, как спецификация на XML, де-факто ставшем стандартом на разработку Web-приложений, использование UDDI-механизма для динамической локализации и «подключения» нового ресурса, возможность интеграции компонентов системы, разработанных различными производителями. Кроме того, механизм координированных

атомарных операций предоставляет поддержку структуризации и обеспечения отказоустойчивости комплексной системы, которая может включать в себя такие «не-программные компоненты», как человеческий фактор, финансы, товары и услуги.

Выводы будут неполными, если не сказать о диверсности как о технологии, в значительной мере способствующей решению проблем проектирования программных средств, разработки интерфейса взаимодействия пользователя и программных компонентов. Более того, диверсность может применяться для валидации надежных систем в течение все цикла их проектирования и разработки.

Дальнейшие исследования в этой сфере предполагают выполнения моделирования композитного (интегрированного) web-сервиса. Для решения данной задачи нужно выполнить следующее:

- разработать и обосновать архитектуру трехкомпонентной системы сервис-ориентированной архитектуры, интегрирующей модули ядра (Middleware), управления (Management Tool) и мониторинга (Monitoring Tool);
- построить имитационную и математическую модели композитной системы;
- провести анализ построенной модели как СМО с учетом допущений касательно числа заявок и размеров очереди;
- разметить и рассчитать модель при различных интенсивностях переходов между состояниями.

Такое комплексное моделирование поможет подробнее разобраться с логикой работы композитного web-сервиса, проанализировать возможные состояния системы как в случае нормального функционирования, так и под влиянием внешних воздействий. На основании проведенного анализа будут сделаны выводы об особенностях функционирования системы в зависимости от типов воздействий.

Список литературы

1. Avižienis A., Lapri J-C., Randel B. Fundamental Concepts of Dependability, UCLA CSD Report no.010028, LAAS Report no.01-145, Newcastle university Report no. CS-TR-739, 2002, 20 p.

2. Харченко В.С. От безотказности электронных устройств к гарантоспособности web-систем: эволюция парадигм, методов и свойств. // Контрольно-измерительные приборы и автоматика, 2004, №9. – С. 4-10.

3. Dobson G., Hall S., Sommerville I. A Container-Based Approach to Fault Tolerance in Service-Oriented Architectures, International Conference of Software Engineering (ICSE), 2005.

4. Kaâniche M., Kanoun K., Martinello M. A User-Perceived Availability Evaluation of a Web Based Travel Agency // Proc. of IEEE Computer Society 2003 International Conf. on Dependable Systems and Networks (DSN 2003), 22-25 June 2003. – San Francisco, CA, USA, 2003. – P. 125-134.

5. Aghdaie N., Tamir Y. Client-Transparent Fault-Tolerant Web Service // Proc. of the 20th IEEE International Performance Computing and Communication Conf., April 2001. – Phoenix, Arizona, 2001. – P. 209-216.

6. Aghdaie N., Tamir Y. Fast transparent failover for reliable web services // Proceedings of the International Conf. on Parallel and Distributed Computing and Systems. November 2003. – Marina del Rey, California, 2003. – P. 757-762.

7. Merzbacher M., Patterson D. Measuring End-User Availability on the Web: Practical Experience // International Performance and Dependability Symposium (DSN), Washington DC, June 2002.

8. Chandra S., Chen P.M. Whither Generic Recovery from Application Faults? A Fault Study using Open-Source Software // Proceedings of the 2000 Conference on Dependable Systems and Networks / Symposium on Fault-Tolerant Computing (DSN/FTCS), June 2000.

9. Aghdaie N., Tamir Y. Implementation and Evaluation of Transparent Fault-Tolerant Web Service with Kernel-Level Support // Proc. of the IEEE Intl Conf. on Computer Communications and Networks, Oct 2002. – Miami, Florida, 2002. – P. 63-68.

10. Romanovsky A., Periorellis P., Zorzo A.F. On Structuring Integrated Web Applications for Fault Tolerance // Technical report. University of Newcastle upon Tyne, 2002.

11. Aghdaie N., Tamir Y. Performance Optimizations for Transparent Fault-Tolerant Web Service // Proceedings of the IEEE Pacific Rim Conference on Communications, Computers and Signal Processing, Victoria, Canada, Aug 2003.

12. Uribarne L., Vallecillo A., Alves C., Castro J. A Non-Functional Approach for COTS-components trading // Proc. of the IV Workshop on Requirements Engineering. Buenos Aires, Argentina. November 22-23, 2001.

13. Andersony S., Dobsonz G., Hallz S., Hughes C. Dependable Grid Services: DIGS Initial Implementation Notes // Proceedings of DIRC April 2004 Workshop, P. 12-26, Newcastle.

14. Desmatre Y., Kanoun K., Laprie J-C. Diversity against Accidental and Deliberate Faults // Proc. Computer Security, Dependability and Assurance: From Needs to Solutions, York, England and Washington, D.C., USA, 1998.

15. Fujita S. Dynamic Collaboration of Businesses Using Web-services // NEC Journal for advanced Technology. – Vol. 1, No. 1. – P. 36-42.

16. Gashi I., Popov P., Stankovic V., Strigini L. On designing dependable service with off-the-shelf SQL servers // Proc. of the Workshop on Software Architectures for Dependable Systems (WADS), ICSE 2003: Portland, Oregon, USA. – P. 191-214.

17. Littlewood B., Strigini L. Redundancy and Diversity in Security. // In Proceedings of the ESORICS 2004, 9th European Symposium on Research in Computer Security, Sophia Antipolis, France, September, 2004. – P. 423-438.

18. Scott D., Sharp R. Specifying and Enforcing Application-Level Web Security Policies // IEEE Transactions on Knowledge and Data Engineering, Vol. 15, No. 4, Jul/Aug 2003.

19. Ferguson D.F., Storey T., Lovering B., Shewchuk J. Secure, Reliable, Transacted Web Services: Architecture and Composition. IBM Corporation. MS and IBM. Technical Report, September, 2003.

20. Боярчук А.В. Обеспечение отказоустойчивой работы web-приложений // Тезисы 4-й конф. ИКТМ. – Х.: НАКУ «ХАИ», 2003. – С. 34-35.

21. Харченко В.С., Горбенко А.В., Боярчук А.В. Безопасность как интегральный атрибут надежности Web-сервисов // Материалы 2-й межд. конф. «ИТ в машиностроении». – Алушта, 6-9 сентября, 2004. – С. 15-17.

Поступила в редакцию 20.07.2006

Рецензент: д-р техн. наук, проф. В.М. Илюшко, Национальный аэрокосмический университет им. Н.Е. Жуковского «ХАИ», Харьков.