9. Koh, Y. S. Rare Association Rule Mining and Knowledge Discovery [Text] / Y. S. Koh, N. Rountree. – New York : Information Science Reference. – 2009. – 320 p.
10. Zadeh, L. Fuzzy sets [Text] / L. Zadeh // Information and Control. – 1965. – № 8. – P. 338–353.
11. Субботін, С. О. Неітеративні, еволюційні та мультиагентні методи синтезу нечіткологічних і нейромережних моделей: монографія [Текст] / С. О. Субботін, А. О. Олійник, О. О. Олійник ; під заг. ред. С. О. Субботіна. – Запоріжжя : ЗНТУ, 2009. – 375 с.
12. Encyclopedia of artificial intelligence [Text] / Eds.: J. R. Dopico, J. D. de la Calle, A. P. Sierra. – New York : Information Science Reference, 2009. – Vol. 1–3. – 1677 p.
13. Интеллектуальные информационные технологии проектирования автоматизированных систем диагностирования и распознавания образов : монография [Текст] / [С. А. Субботин, Ан. А. Олейник, Е. А. Гофман, С. А. Зайцев, Ал. А. Олейник под ред. С. А. Субботина]. – Харьков : ООО "Компания Смит", 2012. – 317 с.
14. Прогрессивные технологии моделирования, оптимизации и интеллектуальной автоматизации этапов жизненного цикла авиадвигателей : монография [Текст] / [А. В. Богуслаев, Ал. А. Олейник, Ан. А. Олейник, Д. В. Павленко, С. А. Субботин под ред. Д. В. Павленко, С. А. Субботина]. – Запорожье : ОАО «Мотор Сич», 2009. – 468 с.
15. Гибридные нейро-фаззи модели и мультиагентные технологии в сложных системах : монография [Текст] / [В. А. Филатов, Е. В. Бодянский, В. Е. Кучеренко и др. под общ. ред. Е. В. Бодянского]. – Дніпропетровськ : Системні технології, 2008. – 403 с.
16. Айвазян, С. А. Прикладная статистика: Исследование зависимостей [Текст] / С. А. Айвазян, И. С. Енюков, Л. Д. Мешалкин. – М.: Финансы и статистика, 1985. – 487 с.
17. Диагностирование нейро-артритических аномалий на основе ассоциативных правил [Текст] / Т. А. Зайко, А. А. Олейник, Н. В. Жихарева, С. А. Субботин // Бионика интеллекта. – 2012. – № 2 (79). – С. 53–57.

В роботі досліджуються питання визначення показників якості для атомарних сервісів в сервіс-орієнтованих системах. Визначено кількісні оцінки показників якості для атомарних сервісів. Запропоновано методи моніторингу та управління сервісами на підставі статистичних даних показників якості сервісів. У статті описані методи вибору екземплярів атомарних сервісів з однаковим інтерфейсом із пулу сервісів

Ключові слова: web-сервіс, SOA, якість, час відгуку, доступність, надійність, якість послуг

В работе исследуются вопросы показателей качества для атомарных сервисов в сервис-ориентированных системах. Определены количественные оценки показателей качества для атомарных сервисов. Предложены методы мониторинга и управления сервисами на основании статистических данных по показателям качества сервисов. В статье описаны методы выбора экземпляров атомарных сервисов, предоставляющих с одинаковым интерфейсом из пула сервисов

Ключевые слова: web-сервис, SOA, время отклика, доступность, надежность, качество услуг

УДК 004.052

МЕТОД ОЦЕНИВАНИЯ ПОКАЗАТЕЛЕЙ КАЧЕСТВА WEB-СЕРВИСОВ

О. В. Рогов*
E-mail: olehrgf@gmail.com
Т. В. Дуравкина
Кандидат технических наук, старший преподаватель*
E-mail: stv_@list.ru
А. Г. Морозова
Кандидат технических наук, старший преподаватель*
E-mail: a.morozova@karazin.ua
*Кафедра теоретической и прикладной информатики
Харьковский национальный университет им. В. Н. Каразина
пл. Свободы, 4, г. Харьков, Украина, 61022

1. Введение

В настоящее время успех бизнеса сильно зависит от того, насколько он автоматизирован и как быстро компания может предложить новую услугу или продукт на рынок.

Практически перед любым IT подразделением компании всегда стоит задача бесперебойного предоставления IT сервисов бизнесу. Реализация традиционных решений для интеграции прикладных программ - непростая задача, требующая существенных капиталовложений. Кроме того, часто при внедрении

необходимо написание программного кода. В связи с этим возникла проблема разработки технологий более быстрой и менее дорогой интеграция приложений. SOA предусматривает размещение сервисов в сети в режиме исполнения, т.е. позволяет автоматизировать эти ресурсоемкие процессы, благодаря чему существенно сокращаются все расходы на интеграцию [1 – 3].

Также, к основным предпосылкам появления SOA можно отнести высокую динамику современного бизнеса и неуклонно возрастающие требования к постоянной адаптации информационных систем по отношению к этой динамике. Уже недостаточно, чтобы информационная система обеспечивала простую автоматизацию информационных и расчетных задач бизнеса. Необходимо стремиться к тому, чтобы быстро меняющиеся условия бизнеса, возникающие вследствие ужесточения конкуренции, находили полное отражение в информационной системе, то есть корпоративная информационная система должна меняться столь же быстро, сколь быстро меняются требования бизнеса и бизнес-процессы компании.

Интеграция разнородных и распределенных данных не в состоянии разрешить все вопросы управления предприятием [4 – 5]. В соответствии с *процессным подходом* наибольшую ценность представляют не сами по себе данные, а использование информации в тех или иных бизнес-процессах компании. В самых современных ИС принято рассматривать как «атомарную» единицу не данные в «чистом» виде, а некоторый сервис, соответствующий какому-то элементарному бизнес-процессу. В частности, такой сервис может просто выдавать какие-то данные, являясь аналогом «*атомарной*» единицы классических ИС.

SOA понимается как *парадигма* организации и использования распределенного множества функций, которые могут контролироваться различными владельцами. Базовым понятием в такой архитектуре являются «информационная услуга».

Информационная услуга (сервис) – это атомарная прикладная функция автоматизированной системы, которая пригодна для использования при разработке приложений, реализующих прикладную логику автоматизируемых процессов, как в самой системе, так и для использования в приложениях других автоматизированных систем [6].

В основе сервис ориентированных архитектур лежат распределенные программные компоненты, предоставляемые или используемые независимыми сторонами. Поскольку доступ к таким компонентам не ограничен рамками организаций, он должен поддерживаться явными контрактами компонентов и общепринятыми стандартами. При этом не менее важно для пользователя, чтобы сервис обеспечивал не только необходимую функциональность, но и другие нефункциональные требования (скорость работы, безопасность и др.). Таким образом, в настоящее время все большее внимание уделяется политике обеспечения качества услуг (quality of service, QoS) [7]. К показателям качества Web-сервисов можно отнести обеспечения необходимого уровня безопасности, надежности и отказоустойчивость.

На обеспечение уровня безопасности Web-сервисов направлены многочисленные стандарты и поддерживающие их методы: WSSecurity, WS-Trust, Extensible Access Control Markup Language (XACML), Security Assertion Markup Language (SAML) и т.д [8]. Например, в среде Internet можно безопасно пользоваться услугами банков, платить по счетам и совершать покупки.

Исследования в областях Web-сервисов и SOA в настоящее время концентрируются на протоколах, функциональности, транзакциях, онтологиях, композиции, семантическом Web и интероперабельности, а надежности и доверительности сервисов уделяется недостаточное внимание.

SOA позволяет разработчикам приложений производить поиск сервисов и использовать сервисы, обеспечиваемые различными поставщиками. Поскольку брокеры сервисов не несут ответственность за качество сервисов, их достоверность не гарантируется. Традиционные методы обеспечения надежности – доказательство корректности, отказоустойчивость, формальная верификация на основе моделей (model checking), тестирование, оценка и т.д. – могут повысить уровень доверительности отдельных сервисов. Однако разработчикам необходимо переработать эти методы для обеспечения возможности их применения к динамическим приложениям, компонуемым из сервисов во время выполнения.

Поэтому актуальной является задача разработки методов управления сервисами с целью обеспечения гарантированного качества обслуживания в SOA системах.

2. Обзор SOA систем

Oracle Fusion Middleware (OFM) [9] состоит из нескольких программных продуктов корпорации Oracle. OFM охватывает несколько услуг, в том числе средств разработки Java EE, средства интеграции, совместной работы и управления содержимым. OFM основан на открытых стандартах, таких как BPEL, SOAP, XML и JMS.

Oracle Fusion Middleware предоставляет программное обеспечение для разработки, развертывания и управления сервис-ориентированной архитектурой.

IBM WebSphere [10] - линейка продуктов предназначена для интеграции и поддержки SOA архитектуры от кампании IBM. Здесь, с точки зрения управления сервисами, основным продуктами является: WebSphere Business Monitor и Tivoli Composite Application Manager для SOA.

WebSphere Business Monitor осуществляет мониторинг бизнес-процессов в реальном времени, обеспечивая наглядное отображение их статуса. WebSphere Business Monitor интегрируется с WebSphere Business Modeler (применяемым при моделировании), так что бизнес-параметры (например, затраты, доходы, производительность) бизнес-процессов, смоделированных в WebSphere Business Modeler, можно было отслеживать в WebSphere Business Monitor и проверять на совместимость.

Tivoli Composite Application Manager для SOA. Появление в многоуровневой архитектуре приложений уровня абстракции сервисов возникает потребность в механизме для мониторинга, управления и измерения

сервисов, которые являются основополагающими конструкциями SOA. IBM Tivoli Composite Application Manager для решений SOA (ITCAM для SOA) обеспечивает отслеживание Web-сервисов на всех уровнях для обнаружения и устранения неисправностей, которые могут возникнуть при вызове и выполнении сервисов.

Microsoft BizTalk Server - это интеграционный пакет, обеспечивающий обмен данными в разных форматах, преобразование данных, создание и обеспечение работы бизнес процессов. Основные направления использования BizTalk Server таковы:

• интеграция приложений внутри организации (Enterprise Application Integration);

• интеграция приложений бизнес-партнеров в рамках структуры B2B;

• автоматизация бизнес-процессов, протекающих в интегрируемых приложениях.

BizTalk Server работает только в среде Windows и в качестве хранилища требует Microsoft SQL Server.

Программное обеспечение *Red Hat JBoss Enterprise SOA Platform* [11] – это эффективная и выгодная платформа интеграции приложений и сервисов, обработки бизнес-событий и автоматизации рабочих процессов. Red Hat JBoss Enterprise SOA Platform задействует методологии текущей и новой интеграции, существенно усовершенствуя качество и скорость выполнения бизнес-процессов. Red Hat JBoss Enterprise SOA Platform, будучи сервисной шиной предприятия и системой автоматизации бизнес-процессов, реализует высокую гибкость и скорость ответов на запросы в открытой корпоративной платформе.

Решение предлагает простой в использовании пакет интеграции с сервис ориентированной архитектурой, которая позволяет выстраивать, развертывать, интегрировать, планировать и представлять приложения и сервисы.

Если рассматривать перечисленные выше системы с точки зрения управления сервисами, можно сделать вывод, что они предоставляют обширные средства мониторинга сервисов и оповещение пользователей в различных ситуациях. Однако они не предусматривают управление политиками качества сервисов, а также не дают возможности конечным пользователям получать данные о показателях качества сервисов.

Основная задача SOA - обеспечить совместное функционирование различных слабосвязанных приложений. Такие приложения могут в значительной мере отличаться как функциональностью, так и нефункциональными характеристиками (производительность, надежность, стоимость и др.), которые должны выполняться в течение всего времени предоставления услуги.

Качество обслуживания (QoS) является одним из основных критериев при выборе пользователем того или иного экземпляра сервиса. При этом пользователи обращают одинаковое внимание, как на функциональные, так и нефункциональные требования.

В настоящее время существует несколько методов определения уровня QoS, который может обеспечить экземпляр сервиса. Ниже приведено описание некоторых из них.

Язык WSDL (Web Service Definition Language) – является стандартным механизмом WS-Policy Framework определенным консорциумом OASIS. Так же для этих целей широко используется UDDI (Universal Description, Discovery, Integration).

Использование WS-компонента, определенного OASIS, возможно лишь для вновь создаваемых сервисов. В первую очередь это связанно с тем, что эти компоненты необходимо встраивать в реализацию самого сервиса. В тоже время, уже существует достаточно большое количество сервисов, которые основаны на технологии SOA, но компоненты QoS не используют. Так же, при использовании WS-компонента, QoS остается открытым вопрос модификации сервиса и изменения параметров QoS, поскольку стандарт не регламентирует данный процесс.

Существенным вопросом остается контроль заявленных параметров качества обслуживания, так как в течение времени эти параметры могут изменяться под действием разного рода факторов. Следовательно, может возникнуть ситуация, когда пользователь запрашивает и получает сервис с одними параметрами QoS, а в процессе использования сервиса эти параметры будут изменяться.

Каждый экземпляр сервиса публикует информацию о себе в едином месте – UDDI реестре, используя XML-документ. В этом документе определены мета-типы, отражающие различные аспекты функционирования данного сервиса (область назначения, владелец, правила использования и различные технические параметры (адрес, используемые протоколы, заявленные показатели QoS)). При необходимости получения доступа к сервису заданного типа, выполняется сканирование всех сервисов, зарегистрированных в UDDI реестре, с целью выбора экземпляра удовлетворяющего требованиям пользователя. В данном случае вопрос контроля текущих параметров QoS на соответствие заявленным так же остается открытым. Существуют рекомендации, согласно которым владелец сервиса должен самостоятельно контролировать состояние экземпляра сервиса и вносить изменения в UDDI-реестр. Однако, данные рекомендации не находят практического применения.

Таким образом, достаточно важной является задача определения и контроля уровня QoS, который может обеспечить заданный экземпляр сервиса. Следовательно, возникает необходимость разработки методов, которые позволят выполнять мониторинг состояния экземпляров сервисов и передавать ее системам сетевого управления.

3. Показатели качества для сервисов по стандарту OASIS

Контракт сервиса охватывает как функциональные, так и не функциональные аспекты поведения сервисного компонента. К функциональным аспектам относится бизнес-семантика операций компонента, включая его интерфейс и соблюдаемый им протокол. Не функциональные аспекты включают технические особенности взаимодействия, такие как сериализация данных и протоколы QoS.

Протоколы QoS содержат информацию о предоставлении качества сервиса. Уровень качества сервиса состоит из четырех показателей качества: время отклика, максимальная пропускная способность, доступность и надежность.

Время отклика. Время отклика означает продолжительность времени с момента отправки запроса до момента получения ответа. Время отклика может меняться и зависит от трех типов задержек: задержка клиента, задержки в сети и задержка сервера.

Задержка клиента (*ClientLatency*) – это время задержки, вызванное клиентской частью приложения, во время обработки запроса. Это сумма времени проходящего от момента запроса к клиенту до момента отправки запроса (CL_1) и от момента приема ответа клиентом до момента завершения обработки ответа клиентом (CL_2).

Задержка в сети (*NetworkLatency*) - это время, затраченное на передачу сообщения запроса и сообщения ответа по сети. Это сумма времени, проходящего между событием «Клиент посылает запрос» и событием «Web-сервер услуг получает запрос» (NL_1), и временем, проходящее между событием «сервер отправляет ответ» и событием «клиент получает ответ» (NL_2).

Задержка сервера (*ServerLatency*) - это время обработки запроса на сервере и формирование ответа на сервере. Это сумма времени, проходящего между событием «сервер посылает запрос» и событием «Web-сервис получает запрос» (SL_1), и времени, необходимого для обработки запроса (SL_2), а также времени, проходящего между событием «Web-сервис посылает ответ» и событием «сервер получает ответ» (SL_3).

Три типа задержки и время отклика (*ResponseTime*) могут быть рассчитаны по следующим формулам:

$$\text{ClientLatency} = \text{CL}_1 + \text{CL}_2, \quad (1)$$

$$\text{NetworkLatency} = \text{NL}_1 + \text{NL}_2, \quad (2)$$

$$\text{ServerLatency} = \text{SL}_1 + \text{SL}_2 + \text{SL}_3, \quad (3)$$

$$\text{ResponseTime} = \text{ClientLatency} + \text{NetworkLatency} + \text{ServerLatency}. \quad (4)$$

Пропускная способность. Пропускная способность определяется как максимальное количество запросов, которые может обработать сервис-провайдер за определенный промежуток времени. Пропускную способность можно вычислить по следующей формуле:

$$\text{MaximumThroughput} = \max\left(\frac{\text{numOfRequests}}{\text{measuredTime}}\right), \quad (5)$$

где numOfRequests – количества запросов обработанных сервером за промежуток времени, measuredTime – промежуток времени, в течение которого проводились измерения.

Доступность системы. Доступность является измерением, которое определяет степень доступности Web-сервиса. Доступность можно выразить следующей формулой:

$$\text{Availability} = 1 - \frac{\text{downTime}}{\text{measuredTime}}, \quad (6)$$

где downTime – время, в течение которого сервис не доступен, measuredTime – время, в течение которого проводились измерения.

Надежность. Надежность - это вероятность возвращения ответа после успешного выполнения Web-сервиса, которую можно определить по следующей формуле:

$$\text{Successability} = \frac{\text{a}_{\text{response}}}{\text{a}_{\text{request}}}, \quad (7)$$

где $\text{a}_{\text{response}}$ – количество ответов от сервиса, $\text{a}_{\text{request}}$ – количество запросов к сервису.

4. Метод мониторинга параметров QoS

Как было упомянуто выше, построение SOA систем на базе WS-компонентов не предусматривает использование единого компонента, который бы мог хранить данные о параметрах, в том числе и о параметрах QoS сервисов. Таким образом, в данных системах необходимо либо вводить новые компоненты в состав систем управления, либо предоставлять пользователям возможность самостоятельного мониторинга состояния параметров QoS экземпляров сервисов.

Одним из подходов к мониторингу и управления состоянием экземпляра сервиса является использование централизованного компонента описания параметров QoS для всех экземпляров сервисов, зарегистрированных в системе.

Данный подход предусматривает наличие компонента мониторинга на стороне провайдера. Основной задачей данного компонента является измерение показателей качества и сбор статистики по ним.

В работе предлагается следующая архитектура системы управления.

Для множества атомарных сервисов, предоставляющих одинаковые услуги с одинаковым интерфейсом, стоится пул сервисов, который управляется программным компонентом - proxy-объектом (*Manager*).

Manager решает следующие задачи:

1. Принятие решения о том, какой сервис будет выбран на основе показателей качества, и перенаправление к нему запроса от пользователя.
2. Мониторинг показателей качества атомарных сервисов из пула.
3. Сохранение статистики о показателях качества атомарных сервисов.

Общая схема предлагаемого решения для данного компонента представлена на рис. 1.

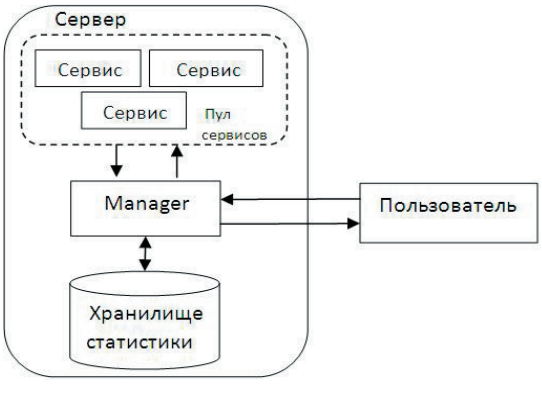

Рис. 1. Общая схема мониторинга показателей QoS для сервисов

Таким образом, конечный пользователь будет обращаться не к конкретному экземпляру сервисов, а к proxy-объекту (*Manager*), с таким же интерфейсом, что и конечные сервисы.

Основной задачей данного менеджера является сбор статистики по показателям качества экземпляров сервисов, и перенаправление запросов от пользователей к ним. Полученные данные по показателям качества сохраняются в хранилище данных. Полученные данные должны использоваться для анализа состояния экземпляра сервиса.

В качестве хранилища данных можно использовать реестр UDDI, компонент tModel которого дополнить атрибутами, описывающими параметры QoS, которые может обеспечить конкретный экземпляр сервиса.

Процесс мониторинга предлагается реализовать в двух режимах: *пассивный мониторинг и активное тестирование*.

В процессе пассивного мониторинга, агент формирует оценки показателей качества, используя статистику обслуживания реальных запросов пользователей сервиса. В данном режиме возможен сбор статистики по следующим параметрам: пропускная способность, время отклика, доступность. Агент собирает статистику и если один из параметров выходит за пределы допустимого, формируется сообщение агенту управления.

В процессе активного мониторинга агент самостоятельно формирует тестовые запросы. Достоинством такого подхода является:

• реальные параметры обеспечиваемого уровня QoS известны до публикации сервиса;

• в случае небольшого количества запросов к сервису, результаты, полученные в ходе активного тестирования, более надежны;

• возможность обнаружения отказа экземпляра сервиса до того, как получит отказ реальный пользователь;

В тоже время, процесс активного мониторинга обладает и следующими недостатками:

• тестовые последовательности формируются искусственно, следовательно, существует вероятность того, что не всегда будут обнаружены сбои в работе экземпляра сервиса;

• тестовые запросы к экземплярам сервисов, могут привести к перегрузкам сервисов и сети.

Следовательно, при использовании стратегии активного тестирования ключевым вопросом является процесс формирования тестовых последовательностей, которые максимально покрывают область допустимых значений пользовательских данных и не приводят к перегрузкам в транспортной сети.

Одним из важных моментов при организации такой системы мониторинга и управления является выбор интервалов опроса агентов (TTimer). Важность данного параметра обусловлена тем, что некорректный его выбор может породить значительное количество служебного трафика в системе в ущерб основному, критически чувствительному к задержкам.

Менеджер (Manager) имеет следующие конфигурационные параметры:

• *mode={passive, active}* – режим работы менеджера.

• *getServiceAlgorithm* – метод выбора экземпляра сервиса.

• *TTimer* – частота выполнения тестовых запросов, в режиме активного тестирования.

К функциям Менеджера относятся:

1. Запрос сертификатов качества экземпляров сервиса.

2. Сбор статистики по показателям качества для всех экземпляров сервиса.

3. Мониторинг нагрузки экземпляров сервисов.

4. Перенаправление запросов к конкретным экземплярам сервиса.

Основную сложность в работе Менеджера представляет механизм выбора конкретного экземпляра сервиса.

5. Алгоритмы выбора экземпляра сервиса

Для выбора конкретного экземпляра сервиса Менеджер использует настраиваемый алгоритм выбора сервиса.

Вначале введем следующие определения:

$S_1 = \langle AV_1, SB_1, RT_1, MT_1 \rangle$ – экземпляр сервиса,

$S_1,...,S_n$ – экземпляры сервисов, зарегистрированных у Менеджера,

$WS = (S_1,...,S_2)$ – множество сервисов, зарегистрированных у Менеджера.

Первый алгоритм основывается на выборе менее загруженного экземпляра сервиса путем выбора конкретного экземпляра из множества сервисов, полученного из общего пула, удовлетворяющего заданным показателям качества.

В конфигурацию алгоритма входят следующие параметры:

• AV_{max} – максимальное значение ;

• AV_{min} – минимальное значение Availability;

• SB_{max} – максимальное значение Successability ;

• SB_{min} – минимальное значение Successability ;

• RT_{max} – максимальное значение ResponseTime ;

• RT_{min} – минимальное значение ResponseTime ;

• MT_{max} – максимальное значение MaximumThroughput;

• MT_{min} – минимальное значение MaximumThroughput;

•isWSEmpty:{error, any} – признак того, что делать, если удовлетворяющие условиям сервисы не найдены;

• defaultQoSIndex – приоритетный показатель качества.

Алгоритма состоит из 4 шагов:

1. Строим множество WS′ по следующему правилу: для каждого сервиса, принадлежащего множеству WS, просматриваются показатели качества, и если они удовлетворяют конфигурации Менеджера, то сервис помещается во множество WS′.

2. Если WS′ = ∅, тогда:

a) если isWSEmpty=error, то алгоритм выдает сообщение об ошибке и останавливается;

b) иначе WS′= WS″, где WS″ - множество сервисов с минимальной загруженностью, которая определяется средствами мониторинга работы сервисов.

3. Выполнение сервиса с наименьшей загруженностью из множества WS′.

4. Пересчитать показатели качества для S_i.

Второй алгоритм основывается на выборе менее загруженного экземпляра сервиса путем выбора кон-

кретного экземпляра из множества сервисов, полученного из общего пула, путем отсеивания сервисов, не удовлетворяющих по интегральному показателю качества.

В конфигурацию алгоритма входят следующие параметры:

- $AV_{priority}$ – приоритет значения Availability;
- $SB_{priority}$ – приоритет значение Successability;
- $RT_{priority}$ – приоритет значение ResponseTime;
- $MT_{priority}$ – приоритет значение MaximumThroughput;
- MaxQoSIndex – максимальное значение суммарного показателя качества для сервиса;
- MinQoSIndex – максимальное значение суммарного показателя качества для сервиса;
- isWSEmpty:{error, any} – признак того, что делать, если удовлетворяющие условиям сервисы не найдены;
- defaultQoSIndex – показатель качества, который будет выбран в качестве приоритетного, если пользователь не укажет параметры для поиска экземпляра сервиса.

Алгоритма состоит из 4 шагов:

1. Для каждого $S_i \in WS$ вычислить QoSIndex по следующей формуле:

$$QoSIndex = AV_{priority} * AV_i + SB_{priority} * SB_i + \\ + MT_{priority} * \frac{MT_i}{\max\{MT_1,...,MT_n\}} + \\ + RT_{priority} * \left(1 - \frac{RT_i}{\max\{RT_1,...,RT_n\}}\right). \quad (8)$$

Т.е., во множество WS′ помещаются те сервисы $S_i \in WS$, у которых интегральный показатель качества удовлетворяет заданной конфигурации Менеджера.

2. Если WS′ = ∅, тогда:

a) если isWSEmpty=error, то алгоритм выдает сообщение об ошибке и останавливается;

b) иначе WS′= WS″, где WS″, – множество сервисов с минимальной загруженностью, которая определяется средствами мониторинга работы сервисов.

3. Выполнение сервиса с наименьшей загруженностью из множества WS″.

4. Пересчитать показатели качества для S_i.

6. Выводы

Анализ средств управления QoS в SOA системах показал, что в настоящее время наиболее используемыми являются политики WSDL и UDDI. Однако данные средства основное внимание уделяют группе функциональных показателей качества (описание бизнес-логики сервиса), которые необходимы для организации работы с сервисами. В тоже время, с точки зрения пользователя, более приоритетными являются показатели качества нефункциональной группы (время отклика, доступность, надежность и т.п.). Для реализации управления параметрами QoS нефункциональной группы предложена модификация UDDI реестра, с целью публикации значений соответствующих показателей качества.

Для получения текущих значений показателей качества сервисов предложена модель мониторинга, основанная на host-based подходе.

Предложена архитектура системы мониторинга состояния сервисов, основанная на построении общего менеджера для пула атомарных сервисов с одинаковым интерфейсом.

Литература

1. Channabasavaiah, K. Migrating to a service-oriented architecture [Текст] / K. Channabasavaiah, K. Holley, E.M. Tuggle. – IBM, 2004.
2. Introduction to Service Oriented Architectures [Электронный ресурс]. – Режим доступа: http://searchdatamanagement.techtarget.com/feature/Introduction-to-service-oriented-architecture-What-is-SOA.
3. Маквитти, Л. Архитектура SOA как она есть [Электронный ресурс] / Лори Маквитти / Сети и системы связи. – 2006. – №2. Режим доступа: http://www.ccc.ru/magazine/depot/06_02/read.html?0104.htm.
4. Linthicum, David. Cloud Computing and SOA Convergence in Your Enterprise [Текст] / David Linthicum. – Boston: Addison-Wesley, 2010.
5. Биберштейн, Н. Компас в мире сервис-ориентированной архитектуры (SOA): ценность для бизнеса, планирование и план развития предприятия [Текст] / Н. Биберштейн, С. Боуз, К. Джонс и др. - М.: КУДИЦ-ПРЕСС, 2007.
6. Граничин, О. Н. Сервисно-ориентированная архитектура ИС ВШМ СПбГУ и проблемы стохастической оптимизации [Электронный ресурс] / О. Н. Граничин, И. Л. Широнов. Режим доступа:http://www.math.spbu.ru/user/gran/sb3/gran_sheron.pdf.
7. Franch, X. Quality of Service (QoS) in SOA Systems [Электронный ресурс]. – 2009. – Режим доступа: http://upcommons.upc.edu/pfc/bitstream/2099.1/7714/1/Master%20thesis%20-%20Marc%20Oriol.pdf.
8. Гладцын, В.А. Сервис-ориентированная архитектура. Стандарты, алгоритмы, протоколы [Текст] / Гладцын В.А., Кринкин К.В., Яновский В.В. – СПб.: СПбГЭТУ «ЛЭТИ», 2006.
9. Oracle Fusion Middleware [Электронный ресурс]. – Режим доступа: http://www.oracle.com/ru/products/middleware/overview/index.html.
10. Программное обеспечение WebSphere [Электронный ресурс]. – Режим доступа: www.ibm.com/software/ru/websphere.
11. JBOSS enterprise SOA platform (ESB) [Электронный ресурс]. – Режим доступа: http://www.redhat.com/resourcelibrary/datasheets/JBoss-SOA-datasheet.