У роботі описаний метод інтелектуалізації платформ для Cloud Computing на основі онтологічного підходу до подання активних інформаційних ресурсів. Запропоновано модель інтелектуальних web-сервісів, які можуть вбудовуватися в програмне забезпечення автоматичним способом

Ключові слова: SEAP-платформа, інтелектуальний web-сервіс

В работе описан метод интеллектуализации платформ для Cloud Computing на основе онтологического подхода к описанию активных информационных ресурсов. Предложена модель интеллектуальных web-сервисов, которые могут встраиваться в программное обеспечение автоматическим способом

Ключевые слова: SEAP-платформа, интеллектуальный Web-сервис

Method of cloud computing intellectualization is presented in the article. It is based on ontological approach to active information resources representation. Intelligent web-service model is introduced. They could be embedded into software automatically

Keywords: SEAP-platform, intelligent web-service

УДК 519.7:007.52

МЕТОД ИНТЕЛЛЕКТУАЛИЗАЦИИ ПЛАТФОРМ ДЛЯ CLOUD COMPUTING

А.Ю. Шевченко
Кандидат технических наук, доцент
Кафедра искусственного интеллекта*
Контактный тел.: (057) 701-33-60, 050-61-92-638
E-mail: shevchenko@vk.kh.ua

Е.Л. Шевченко
Кандидат технических наук, доцент
Кафедра программного обеспечения ЭВМ*
Контактный тел.: (057) 701-33-60, 050-57-48-372
E-mail: olena.l.shevchenko@gmail.com
*Харьковский национальный университет радиоэлектроники
пр. Ленина, 14, г. Харьков, Украина, 61145

1. Введение

Как показывает практика, к настоящему моменту рынок продажи услуг web-сервисов – концепция Software as a Service (SaaS) – накопил большой потенциал. Его востребованность обоснована многими факторами: выполнение сложных, однотипных для целого класса приложений подзадач при сервис-ориентированной архитектуре приложения переносится на надежный, удаленный интернет-сервер и не требует больше от потребителя обладания мощными аппаратными средствами. Качество сервиса гарантируется имиджем компании-создателя и числом уже имеющихся у него клиентов. Конечные разработчики приложений благодаря наличию разнотипных, доступных через интернет удаленных модулей способны создавать мощные приложения в кратчайшие сроки с минимальными усилиями. При этом разработчикам web-служб как поставщикам услуг больше не нужно тратить средства на производство носителей с распространяемой продукцией, поиск каналов их сбыта и методик распространения обновлений. Эта концепция получила название Cloud Computing [1], а число желающих продавать свои информационные и вычислительные услуги как SaaS с каждым годом все возрастает. Примерами удачных SaaS проектов в настоящий момент являются Google Maps API, Google Chart Tools,Yandex Speller и др. По прогнозам аналитиков Gartner group к 2015 г. cloud computing станет предпочтительным способом реализации ИТ-услуг в крупнейших компаниях мира [2].

В данной статье излагается подход к совершенствованию технологии cloud computing, который основывается на предложенной в работе [3] идее интеллектуализации платформы для развертывания и выполнения облачных приложений. В систему знаний платформы предлагается включить не только сведения о её собственной структуре, конфигурации и состоянии, как сделано в [3]. Кроме этого, с момента публикации приложения на платформе и на протяжении всего периода его эксплуатации платформа будет аккумулировать знания о его назначении, качестве, условиях использования, отзывах клиентов. Таким образом, кроме стандартных возможностей, которые ранее предназначались для удовлетворения нужд прежде всего разработчиков облачного приложения, усовершенствования, предлагаемые в данной работе, направлены на организацию знаний, необходимых для успешной популяризации опубликованных на ней приложений, гарантирования их надежности, автоматизации поиска и вызова необходимой службы по её высокоуровневому описанию. Такой подход призван способствовать формализации и систематизации знаний про облачные приложения аналогично тому, как концепция Semantic Web предназначалась изначально для формализации и упорядочивания знаний про ресурсы всемирной паутины со всеми вытекающими из этого следствиями.

2. Анализ предметной области и постановка задачи

Идея создания единой среды – Service-Enabled Application Platform (SEAP) – для разработки и функционирования облачных приложений стала новым витком в развитии cloud computing. В своем текущем варианте SEAP предоставляет широкий выбор виртуальных платформ для запуска web-служб, надежные распределенные хранилища данных, шаблоны для разработки стандартных типов служб и их модулей, универсальное API для доступа к сервисам хранилища, для решения задач аутентификации, загрузки данных по URL, отправки электронной почты, использования планировщика задач и пр. Также SEAP позволяет совместно использовать гигантские вычислительные кластеры нескольким облачным приложениям. Существование SEAP избавляет разработчика SaaS от рутинных забот по организации средств размещения своего приложения, усилий по обеспечению его надежности, масштабируемости, позволяя сконцентрировать внимание на реализации основной идеи. На данный момент три основных конкурента на рынке SEAP – это Google App Engine, Amazon Web Services и Windows Azure.

Активные исследования по возможностям усовершенствования концепции SEAP продолжаются. Её интеграция с парадигмой автономных вычислений [4] и концепцией Global Understanding Environment (GUN) [5] привела к рождению идеи об интеллектуальной платформе, базирующейся на слое проактивных самоосознающих агентов [3]. Эта модификация находится над верхним уровнем стека интерфейсов классической архитектуры cloud computing. Каждая функция или ресурс такой интеллектуализированной SEAP (будь это интерфейс доступа к хранилищу данных, служба отправки почты или менеджер распределения нагрузки на многопроцессорную систему) являются интеллектуальным проактивным агентом, способным к взаимодействию с себе подобными, обладающим возможностью автоматически планировать поведение, отслеживать и корректировать свое состояние, основываясь на собственной роли в бизнес процессе, использовать знания других агентов. Целью каждого из таких агентов является оптимизация отдельных критериев работы платформы: уменьшение времени отклика, обеспечение отказоустойчивости, адаптация к изменяющейся конфигурации и появлению новых возможностей SEAP платформы и пр.

Создавать службы, основывающие свою работу на таком промежуточном интеллектуальном слое, проще, чем для классической SEAP; кроме того разработчики повышают стабильность работы своих облачных приложений, снимают необходимость модифицировать их при изменении конфигурации платформы, в т.ч. достигается переносимость приложения при смене SEAP платформы (что на данный момент практически неосуществимо) или модели данных в хранилище.

Как видно из проведенного обзора, на данный момент, все перечисленные SEAP-платформы сконцентрировали свои усилия на борьбе за предпочтения разработчиков SaaS и удовлетворении их потребностей наилучшим образом. Когда же платформа достигнет поставленной цели и её клиентами станут сотни и тысячи облачных приложений весомую составляющую в рейтинге, а значит и в востребованности платформы станет играть еще один компонент, на данном этапе остающийся для авторов SEAP в тени. Это удобство использования, доверие и мнение конечных потребителей – клиентов SaaS приложений. Количество услуг, предоставляемых функционирующими на платформе приложениями, будет расти и встанет вопрос об упорядочении и управлении знаниями об этих услугах аналогично тому, как вот уже около 10 лет активно развиваются методики упорядочивания и формализации знаний в сети интернет.

В следующем разделе приводятся ключевые моменты интеллектуальной SEAP, поддерживающей ориентацию на удовлетворение потребностей конечных пользователей облачных приложений, а далее излагается методика их реализации. Для упрощения изложения материала такая архитектурно и функционально дополненная платформа далее будет называться UFoCuP (User Focused Cloud Platform).

3. Краткая характеристика UFoCuP

Предложенный в работе [3] модифицированный стек облачных вычислений приведен на рис. 1. Управляемый агентами фреймворк замещает для облачного приложения API классической SEAP платформы.

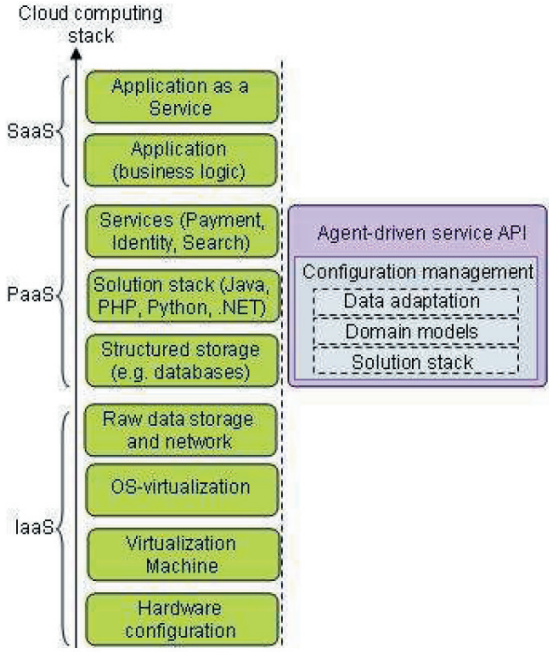

Рис. 1. Место управляемого интеллектуальными агентами API в стеке облачных вычислений

UFoCuP встраивается в стек облачных вычислений аналогичным образом (рис. 2).

Как видно из рис. 2, сервисы UFoCuP находятся для конечного пользователя на уровне обслуживаемых им SaaS-приложений и при желании могут даже заместить их. Т.е. пользователь имеет возможность запросить у платформы выполнение некоторой услуги, не указывая, какой из расположенных на ней сервисов должен обработать запрос. В этом случае платформа сама выбирает исполнителя по нескольким критериям, которые будут описаны далее.

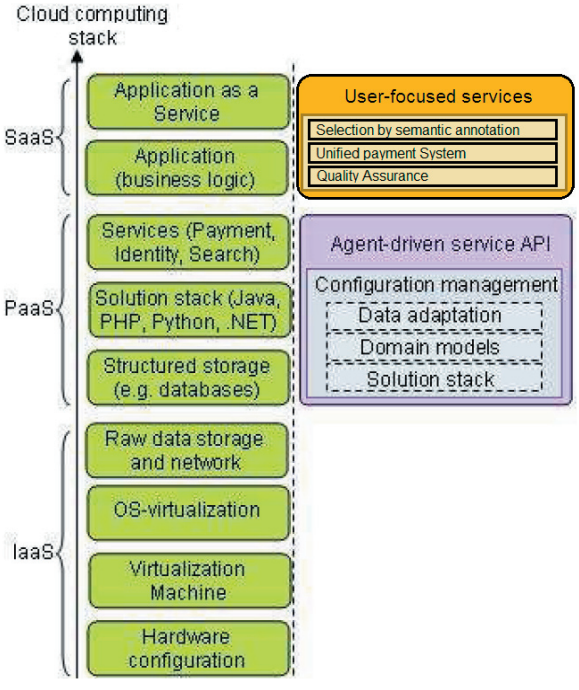

Рис. 2. Стек облачных вычислений при использовании UFoCuP

Основные новые задачи UFoCuP приведены в табл. 1.

Таблица 1

Задачи UFoCuP в контексте взаимосвязи с пользователем и опубликованным на платформе службами

Ориентированные на **конечного пользователя**	Ориентированные на **облачную службу**
1	2
1) возможность поиска службы по её семантическому описанию (популяризация)	
2) возможность обращения к службе без указания конкретного идентификатора, а основываясь только на семантическом описании желаемой услуги	
3) единая система аутентификации пользователей	
4) единообразный интерфейс, предоставляемый всеми расположенными на платформе облачными приложениями	
5) стандартизированная система оплаты услуг за использование приложения	6) получение процента от прибыли службы при использовании её пользователем в соответствии с выбранной системой оплаты
7) доступность для клиента всех предыдущих версий приложения несмотря на наличие обновлений	
8) гарантия качества работы опубликованных сервисов, в т.ч. соответствия заявленных сервисом характеристик фактическим	
9) ведение статистики использования службы и её рейтинга	
10) возможность интеграции с популярными средами разработки (такими как Eclipse, Microsoft Visual Studio и пр.)	

Далее приводятся методики реализации перечисленных задач UFoCuP.

4. Методика поиска и вызова необходимой службы, среди служб, развернутых на платформе, по семантическому описанию

Для обеспечения возможности поиска службы по описанию, близкому к естественному, предлагается привязывать к службе семантическую аннотацию. Ранние работы, посвященные методам компарации программных компонент, отдавали предпочтение сравнению их сигнатур [6] и спецификаций, представленных как набор ограничений на методы и свойства программного элемента: инварианты, пред/пост условия выполнения методов [7]. Этот подход эффективен для описания низкоуровневого поведения компонента, но слишком громоздок для формализации его назначения и возможностей. Известен также простой подход к описанию веб-службы набором ключевых слов, как предлагается в работе [8]. В этом случае поиск по такому описанию должен вестись чисто статистическими методами, наподобие TF*IDF, со всем присущими им недостатками. В данной работе для описания, а, следовательно, поиска по запросу клиента предлагается использовать метод семантического аннотирования ресурса, основанный на технологии Semantic Web. Данный подход является продолжением предложенной в работах [9,10] методики к описанию профиля веб-сервиса на основе онтологической модели OWL-S (профиля OWL для описания веб-сервисов). В этой модели значительное внимание уделяется аннотации параметров сервиса и входящих в него процессов, но недостаточно развит раздел, описывающий категорию, к которой относится веб-сервис. Описание и поиск сервиса на основе расширенной аннотации его категории дополнит существующую модель и позволит значительно упростить и ускорить выполнение наиболее важных и часто решаемых UFoCuP задач 1 и 2 (табл. 1).

Предлагаемая структура аннотации – это набор триплетов «объект-атрибут-значение», формализующих совершаемое сервисом действие и объект, над которым это действие выполняется. Структура аннотации приведена на рис. 3.

Такая структура аннотации упрощенно формализует отношения между частями простого предложения (верхний ряд примитивов на рис. 1). Пример аннотации сервиса по «Информированию о географическое местоположении объекта наблюдения» приведен на рис. 4.

Для поиска службы по такому семантическому описанию предлагается запрос пользователя строить как аннотацию аналогичного вида и отбирать и упорядочивать все предоставляемые платформой службы путем вычисления степени семантической эквивалентности двух программных компонентов по их аннотациям. Степень семантической эквивалентности может быть выражена с помощью одной из метрик расстояния между двумя RDF графами, вычисляемых на основе сравнения использованных в них понятий, определенных в единой онтологии задач.

Аналогичные аннотации, но прикрепленные к параметрам веб-службы, позволят не только детализировать критерии поиска, но и вызывать службу на основе её OWL-S описания. С их помощью в момент вызова становится возможным сопоставлять параме-

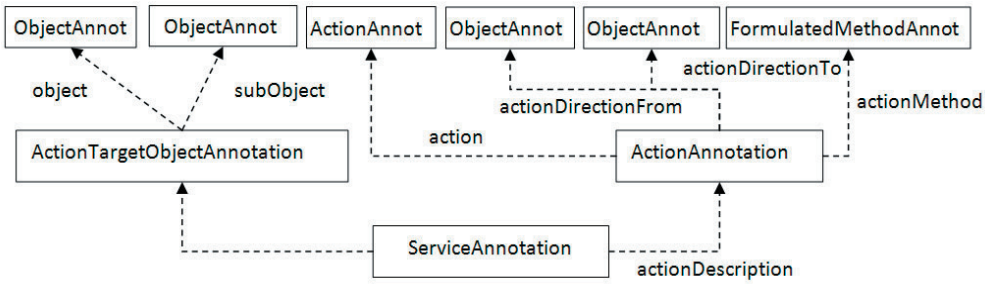

Рис. 3. Схематическая структура аннотации web-сервиса

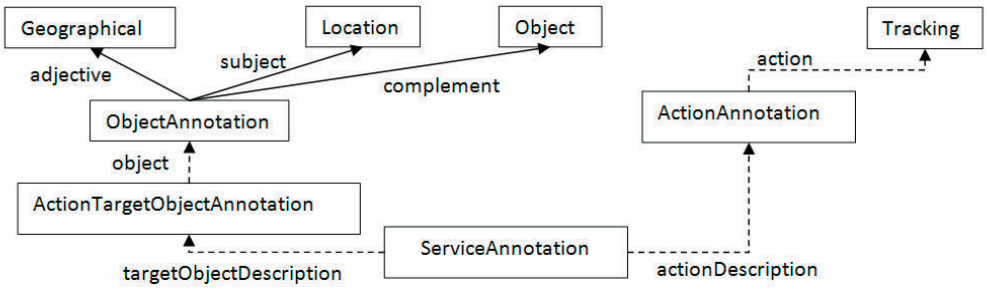

Рис. 4. Пример экземпляра аннотации сервиса

тры, передаваемые клиентом в службу, с ожидаемыми ею параметрами. При этом больше нет необходимости привязываться при сопоставлении к таким негибким критериям, как имя или порядковый номер параметра.

5. Методики оплаты услуг за использование службы

Существует общепринятый метод определения стоимости использования web-сервисов. Суть его заключается в том, что за каждое использование web-сервиса предлагается взимать очень маленькую плату со счета потребителя услуг. Такой подход удобен, но имеет некоторые недостатки в случае, если разработчиком был сделан программный продукт и туда был встроен web-сервис, а затем этот программный продукт был передан заказчику. В этом случае заказчик должен с течением времени пополнять счет, с которого взимает оплату используемая его приложением web-служба. Значительно более перспективным подходом в этом случае может стать оплата за встраивание web-сервиса в каждую единицу продукции. В этом случае разработчик оплачивает услуги web-сервиса на весь период эксплуатации программного продукта, но для каждого его экземпляра. Таким образом cloud-платформа сможет гарантировать длительную поддержку каждого конкретного экземпляра её приложения, значительно превышающую срок его службы.

6. Методики обеспечения гарантий качества опубликованных на платформе служб

Для решения этой задачи предлагается выдвинуть обязательное к исполнению требование для размещаемой на платформе службы: каждый разработчик совместно с публикуемой службой предоставляет набор тестов, выполнение которых, по его мнению, гарантирует её работоспособность.

В качестве тестов в упрощенном варианте может выступать набор запросов и результатов их выполнения. В более сложном варианте, когда ответ на тест неоднозначен или не может быть проверен тривиально, разработчик дополнительно к тестовым входам предоставляет анализатор результатов работы сервиса, код ответов которого показывает, считать ли полученный выход корректным.

После развертывания перед окончательной публикацией служба проходит начальное тестирование на тестах разработчика.

В случае поступления претензий экономических, по безопасности или целостности данных со стороны пользователей, работа службы приостанавливается до выяснения обстоятельств.

7. Методика ведения рейтинга служб

Это самая простая из перечисленных задач. Она должна быть реализована как средство обратной связи с пользователем, который в режиме голосования отдает службе баллы. Бал службы может формироваться на основе добровольных отзывов пользователя после эксплуатации (субъективная составляющая) и фонового сбора информации о количестве обращений к службе: количестве первичных обращений, количестве повторных обращений – (объективная составляющая). Объективная составляющая рейтинга службы может быть рассчитан по формуле:

$$R_{f^i} = \frac{C\_2_{f_i}}{C\_all_{f_i}},$$

где:

f_i – служба, для которой рассчитывается рейтинг;

C_2 – количество повторных обращений к службе;

C_all – полное количество обращений к службе.

Выводы

Описанные выше функциональные возможности, ориентированные на систематизацию знаний о

размещенных на платформе службах и удовлетворении прочих потребностей конечных пользователей, могут показаться на данный момент второстепенными. На самом деле, этап становления cloud computing рано или поздно закончится, перечень технических задач, предоставляемых платформами в помощь функционирующим на них облачным приложениям, станет более или менее стандартизированным. В этот момент на популярность, а следовательно, успешность платформы, в большей степени начнут влиять факторы, описанные в статье. Они предоставят пользователю гарантии качества развернутых на платформе услуг, позволят легко находить службы, выполняющие требуемые ему операции. Если таких служб будет несколько, то помогут сделать выбор в пользу более надежной или дешевой службы на основе субъективных критериев выбора. Для разработчиков служб хостинг на такой платформе заменит дорогостоящую и трудоемкую рекламу. Все это позволит интеллектуализированным SEAP-платформам значительно выиграть перед своими конкурентами.

В данной статье предложено решение проблемы автоматического выбора web-сервисов интеллектуальными системами для в достижении поставленных целей. Предложенный в этой статье метод интеллектуализации платформ для cloud computing дает необходимую информацию интеллектуальным системам для применения активных ресурсов. А модель интеллектуальных web-сервисов показывает, как можно составить семантический профиль активных ресурсов.

Литература

1. Hayes, B. Cloud Computing [Текст] / B. Hayes // Communications of ACM – 2008. – Vol. 51, No 7. – P. 9-11.
2. Gartner: Seven years needed for mass cloud computing acceptance [Электронный ресурс] / Velum Media Portal. – Режим доступа: \WWW/ URL: http://www.tgdaily.com/trendwatch-features/41306-gartner-seven-years-needed-for-mass-cloud-computing-acceptance – 03.05.2011 г. – Загл. с экрана.
3. Nikitin, S., Terziyan, V. Mastering intelligent clouds. Engineering Intelligent data processing services in the cloud [Текст] / S. Nikitin, V. Terziyan // ICINCO – 2010. – V. 1. – P. 174-181.
4. Kephart, J.O. Chess, D.M. The vision of autonomic computing [Текст] / J.O. Kephart, D.M. Chess // IEEE Computer. – 2003. – Vol. 36(1). – P. 41-50.
5. Terziyan V., Katasonov A. Global Understanding Environment: Applying Semantic Web to Industrial Automation [Текст] / V. Terziyan, A. Katasonov // Emerging Topics and Technologies in Information Systems. – IGI Global, 2009. – P. 55-87.
6. Zaremski, A. M., Wing, J. M. Signature matching: a tool for using software libraries [Текст] / A.M. Zaremski, J. M. Wing // ACM Transactions on Software Engineering and Methodology. – 1995. – Vol 4(2). – P.146–170.
7. Zaremski, A. M., Wing, J. M. Specification Matching of Software Components [Текст] / A.M. Zaremski, J. M. Wing // ACM Transactions on Software Engineering and Methodology. – 1997. – Vol. 6, No. 4. – P. 333-369.
8. Wang, Y., Stroulia, E. Semantic Structure Matching for Assessing Web-Service Similarity [Текст] / Y. Wang, E. Stroulia // ICSOC. – 2003. – P. 194-207.
9. Goscinski, A., Brock, M. Toward dynamic and attribute based publication, discovery and selection for cloud computing [Текст] / A. Goscinski, M. Brock // Future Generation Computer Systems. – 2010. – Vol. 26, Issue 7. – P. 947-970.
10. Klusch, M. OWLS-MX: A hybrid Semantic Web service matchmaker for OWL-S services [Текст]/ M. Klusch , B. Fries, K. Sycara // Web Semantics: Science, Services and Agents on the World Wide Web. – 2009. – Volume 7, Issue 2. – P. 121-133.