Интернет-журнал «Науковедение» ISSN 2223-5167 https://naukovedenie.ru/

Том 9, №6 (2017) https://naukovedenie.ru/vol9-6.php

URL статьи: https://naukovedenie.ru/PDF/107TVN617.pdf

Статья опубликована 17.01.2018

Ссылка для цитирования этой статьи:
Волушкова В.Л. Моделирование качества web-сервисов // Интернет-журнал «НАУКОВЕДЕНИЕ» Том 9, №6 (2017) https://naukovedenie.ru/PDF/107TVN617.pdf (доступ свободный). Загл. с экрана. Яз. рус., англ.

УДК 004.4'242

Волушкова Вера Львовна
ФГБОУ ВО «Тверской государственный университет», Россия, Тверь[1]
Доцент кафедры «Информатики»
Кандидат технических наук
E-mail: w2l@pisem.net
РИНЦ: http://elibrary.ru/author_profile.asp?id=290631

Моделирование качества web-сервисов

Аннотация. Web-сервис – это программный интерфейс, описывающий набор операций, которые могут быть вызваны удаленно по сети посредством стандартизированных XML-сообщений. Web-сервис является единицей модульности при использовании сервис-ориентированной архитектуры приложения.

Для поставщиков сервисов важным аспектом технологии Web-сервисов является качество обслуживания. Сервисы без гарантированного качества могут быть приемлемыми в простых случаях обслуживания. Качество обслуживания является критически важным фактором при выборе Web-сервисов в качестве составляющих для сложных бизнес-процессов.

В данной работе моделируется качество Web-сервисов. Созданная модель качества позволяет не только учесть особенности построения составных сервисов, но и связанную с Web-средой неопределенность в работе сервисов. Для этого показатели качества как атомарных, так и композитных Web-сервисов считаются вероятностными. В вероятностной среде оценка качества составного сервиса не всегда может напрямую вычисляться по оценкам его составных частей. Модель рассматривает пять базовых структурных конструкций, каждая из которых организует сервисы-компоненты уникальным образом.

С помощью предложенной методики можно оценить качество композитного Web-сервиса и гарантировать, что сервис сработает на определенном уровне качества. Это доказано проведенными вычислительными экспериментами, результаты которых представлены в работе.

Ключевые слова: Web-сервис; композитный Web-сервис; качество Web-сервиса; показатели качества; время отклика; надежность; оптимизация; рекурсивный алгоритм; «жадный» алгоритм; организация сервис-компонентов

Web-сервисы – технология, позволяющая приложениям взаимодействовать друг с другом независимо от языка программирования, на котором они написаны, а также от платформы, на которой они развернуты. Web-сервис – программная система,

[1] 170004, г. Тверь, Садовый переулок, 35, ауд. 310а

идентифицируемая строкой URI, чьи общедоступные интерфейсы определены на языке XML. Группа Web-сервисов, взаимодействующая друг с другом, определяет приложение Web-сервисов в рамках сервис-ориентированной архитектуры (Service-Oriented Architecture – SOA).

Для поставщиков сервисов важным аспектом технологии Web-сервисов является качество обслуживания (Quality of Service, QoS). QoS является критически важным фактором при выборе Web-сервисов в качестве составляющих для сложных бизнес-процессов [14].

Целью работы является разработка и тестирование модели качества сервисов, позволяющей не только учесть особенности построения составных сервисов, но и связанную с Web-средой неопределенность в работе сервисов. Для этого показатели QoS как атомарных, так и композитных Web-сервисов считаются вероятностными.

Бизнес процесс может быть представлен как поток сервисов или составной Web-сервис. Термин «составной сервис» обозначает множество запусков Web-сервисов, которое представляется направленным графом.

По мере возрастания популярности Web-сервисов все больше организаций основывает свои ИТ-системы на технологии их использования. В простых случаях приемлемыми оказываются сервисы без гарантированного качества обслуживания. Показатели качества обслуживания (Quality of Service, QoS) является важным фактором при выборе Web-сервисов в качестве составляющих для сложных бизнес-процессов. Поэтому аспекты надежности, доступности и другие аспекты QoS фиксируются в стандартах, в частности, в стандарте Business Process Execution Language for Web-Services (BPEL4WS). Необходимость составлять спецификации QoS для Web-сервисов объясняется двумя обстоятельствами. С одной стороны, клиентам нужно качественное обслуживание, а с другой – поставщики услуг стремятся к достижению оптимального баланса между удовлетворенностью пользователей и уровнем загрузки системы. Например, наиболее важные транзакции, такие как обработка банковского платежа, должны выполняться с высоким приоритетом.

Показатели QoS для Web-сервисов можно разделить на пять категорий: быстродействие, ресурсоемкость, функциональная надежность, свойства транзакционности, безопасностьх[15]. Эти качества реализуются на различных уровнях в приложении: уровне экземпляра класса сервиса, уровне класса сервиса и системном уровне.

В таблице представлены показатели QoS различных категорий, распределенные по уровням организации приложения.

Таблица 1

Показатели QoS и уровни организации приложения

Категория	Уровень экземпляра	Уровень класса	Системный уровень
Быстродействие	Время отклика: время, прошедшее с момента подачи запроса до получения ответа.		Пропускная способность: число выполненных запросов в единицу времени.
Ресурсоемкость	Стоимость: денежные затраты за выполнение сервиса.		Требуемая память, ресурс процессора, ширина канала.
Функциональная надежность	Надежность: вероятность того, что сервис выполнится успешно. Доступность: вероятность того, что сервис может быть вызван.		Время на починку.
Свойства транзакционности		ACID properties – Commit protocol.	
Безопасность		Конфиденциальность. Шифрование.	Шифрование на транспортном уровне.

Составлено авторами

Гарантия качества потока сервисов привлекает внимание разработчиков, а именно способы получить показатели качества для потока из показателей QoS его составных частей. Этот процесс имеет некоторые особенности. Во-первых, в потоках сервисы довольно свободно связаны и нет промежуточного программного обеспечения (типа CORBA, например) для координации и выполнения отдельных сервисов. Кроме того, сервис – черный ящик, и неизвестно заранее работает он или нет и так ли как надо. Это влияет на предположение о том, что QoS потока сервисов может быть получено из значений параметров QoS составляющих его Web-сервисов. В обычных процессах составляющие и их механизмы выполнения фиксируется на этапе проектирования. В обычном приложении сложно, если вообще возможно, динамически выбрать механизм выполнения для отдельного задания в соответствии с требованиями качества.

Напротив, для отдельных сервисов в потоке поставщик сервиса может быть найден в ходе выполнения потока. Эти поставщики обычно разнородны и автономны. Так же, отдельный сервис может иметь несколько поставщиков на выбор. Это значит, что приложение может оптимально выбрать поставщиков отдельных сервисов для соответствия с требованиями QoS.

При существующих подходах к моделированию качества Web-сервисов невозможно с достаточной точностью получить показатели качества для составного сервиса из QoS его составляющих. Точность получаемых показателей качества особенно важна при автоматическом выборе сервиса для составления еще более сложного приложения.

Существуют различные подходы к моделированию качества Web-сервисов.

Ряд работ посвящены оптимальному выбору сервиса для выполнения отдельной операции. Оптимизация потока сервиса и выбора отдельных сервисов для составного с целью обеспечения требуемого уровня качества допускает различные подходы. Можно применять нечеткую логику при выборе Web-сервиса, удовлетворяющего параметрам качества, для выполнения отдельного задания [9]. Рейтинг web-сервисов, основанный на алгоритме поисковой машины поставщика услуг (SPSE) предложен в [10].

Метод критического пути (Crtitical Path method – CPM) или метод оценки и пересмотра планов [7] (program-evaluation review technique – PERT), применяемые в управлении проектами для обнаружения узких мест в плане выполнения проекта, могут также быть применены для построения и анализа хода выполнения составного сервиса. Эти методы позволяют определить последовательность этапов, определяющую минимальное время, требуемое для выполнения всего проекта. При этом время выполнения каждой задачи для метода критического пути имеет фиксированное значение, в методе оценки и пересмотра планов используется бета-распределение и независимость (с точки зрения вероятности) для времени выполнения каждой отдельной задачи.

Большинство работ [1, 3, 4] касается статического случая, полагая, что каждый сервис имеет заранее определенные показатели QoS. Преимуществом такого подхода является упрощение выводов. Например, допустим, что сервис W состоит из последовательно вызываемых сервисов W_1, W_2 и W_3 с временами отклика 2, 3 и 4 соответственно. Очевидно, что время отклика сервиса W равно 9. Такой подход имеет существенные ограничения в применимости.

Время отклика Web-сервиса зависит от таких неопределенных причин как число поступивших запросов к сервису и текущая загруженность сервера, на котором запускается сервис. Очевидно, что доступность и надежность также связаны с неопределенностью. Таким образом, параметры QoS в сущности вероятностны. Каждый показатель QoS Web-сервиса можно представить дискретной случайной величиной с конечной областью определения. Подробно данный вопрос рассмотрен в [13]. Дискретные случайные величины для атомарных

сервисов могут быть получены статистически и использоваться напрямую, без проверки каких-либо критериев согласия и приближения к непрерывным распределениям.

В вероятностной среде QoS составного сервиса не всегда может напрямую получиться из QoS его составных частей. Например, рассмотрим два сервиса W_1 и W_2, и поток W, в котором W_1 и W_2 запускаются параллельно.

Допустим, нужно получить среднее время отклика W. Будет неправильно считать средним временем выполнения W максимум из математических ожиданий времени выполнения W_1 и W_2. Недостаточно учитывать только средние величины, как предполагается в [1]. Можно учитывает только QoS сервисов без требований QoS пользователей и предпочтений по аспектам QoS. Такой подход учитывает ситуацию, когда нет приемлемого решения для выполнения ограничений QoS, установленных пользователями [8]. Так же нереалистично предполагать, что параметры QoS имеют нормальное распределение. Как показано выше, случайная величина времени отклика может иметь более сложную структуру.

Естественно, описывать надежность и стоимость Web-сервиса с помощью функций плотности распределения с областью определения, задаваемой множеством {0(неудача),1(успех)} и множеством возможных стоимостей соответственно. Время отклика также можно смоделировать дискретной случайной величиной, если рассмотреть разбиение области определения ее функции плотности – интервала (0; +∞).

Простой алгоритм разбиения на равные по ширине интервалы может использоваться для трансформации времени отклика в дискретную величину. Пример:

Предположим, что вероятность того, что сервис сработает за время в интервалах [0, 1], (1, 4] и (4, 7] – 0.2, 0.6 и 0.2 соответственно. Это время отклика моделируется случайной дискретной величиной *X*.

Dom(X) = {0, 1, 4, 7}. Функция плотности для *X* задается как:

$$f_x(0) = 0$$
$$f_x(1) = 0.2$$
$$f_x(4) = 0.4$$
$$f_x(6) = 0.2$$

Для того чтобы получить вероятностные параметры QoS для составного сервиса из показателей качества его составляющих нужно использовать действия со случайными величинами, представляющими показатели качества простых сервисов.

Рассмотрим операции со случайными величинами.

Пусть X – дискретная случайная величина с конечной областью определения $Dom(X)$. Пусть Y – независимая дискретная случайная величина с областью определения $Dom(Y)$. Функции плотности распределения этих величин заданы как $f_X(x) \cdot u \cdot f_Y(y)$ соответственно. Пусть Z – дискретная случайная величина, полученная при помощи какой-либо операции над X и Y.

Рассмотрим операцию сложения: Z = X + Y.

Область определения функции состоит из элементов, каждый из которых представляет сумму элементов из множеств $Dom(X)$ и $Dom(Y)$:

$$Dom(Z) = \{z_1, z_2, ..., z_k\},$$
$$\max\{m, n\} \le k \le mn,$$

Каждая $z_i, 1 \le i \le k - сумма \quad некоторых \cdot x \in Dom(X), y \in \quad Dom(Y)$

Функция плотности величины Z:

$$f_Z(z_i) = \sum_{x \cdot + y = z_i} f_X(x) \cdot f_Y(y)$$

Рассмотрим пример:

$$\text{X: } Dom(X) = \{1,2\},\ f_X(1) = 0.3,\ f_X(2) = 0.7$$

$$\text{Y: } Dom(Y) = \{9,10,20\},\ f_Y(9) = 0.4,\ f_Y(10) = 0.4,\ f_Y(20) = 0.2$$

$$Z = X + Y$$

$$Dom(Z) = \{10,11,12,21,22\}$$

$$f_Z(10) = 0.3 \cdot 0.4 = 0.12,$$

$$f_Z(11) = 0.7 \cdot 0.4 + 0.3 \cdot 0.4 = 0.4,$$

$$f_Z(12) = 0.7 \cdot 0.4 = 0.28,$$

$$f_Z(21) = 0.3 \cdot 0.2 = 0.06,$$

$$f_Z(22) = 0.7 \cdot 0.2 = 0.14$$

Функция плотности вероятности случайной величины Z может быть посчитана в два этапа: сначала создается массив из сумм всех элементов областей определения X и Y и считается для них значения функции $f_X(x) \cdot f_Y(y)$, затем полученный массив сортируется и суммируются элементы, для которых аргумент функции плотности одинаков. Сложность расчета при таком подходе будет $O(m \cdot n \cdot \log(m \cdot n))$.

При сложении (и как будет показано ниже, умножении) случайных величин с заданными функциями плотности распределения область определения результирующей случайной величины становится существенно шире, чем области определения исходных величин. В худшем случае $|Dom(Z)| = m^n$.

Аналогично сложению выполняется умножение дискретных случайных величин с конечными областями определения: Z = X * Y:

Область определения функции состоит из элементов, каждый из которых представляет произведение элементов из множеств $Dom(X)$ и $Dom(Y)$:

$$Dom(Z) = \{z_1, z_2, ..., z_k\},$$

$$\max\{m,n\} \le k \le mn,$$

$$z_i, 1 \le i \le k - произведение \quad x \in Dom(X), y \in Dom(Y)$$

Плотность Z определяется по формуле:

$$f_Z(z_i) = \sum_{x \cdot y = z_i} f_X(x) \cdot f_Y(y)$$

Функция плотности вероятности случайной величины Z может быть аналогично операции сложения посчитана в два этапа. Сложность расчета при таком подходе будет также O(mnlog(mn)).

Введем операцию максимума для дискретных случайных величин: Z = max(X,Y).

Область определения Z – объединение областей определения X и Y:

$$Dom(Z) = Dom(X) \bigcup Dom(Y)$$

Функция плотности для Z считается по формуле:

$$f_Z(z_i) = \begin{cases} f_X(z) * \sum\limits_{\substack{y<z_i \\ y \in Dom(Y)}} f_Y(y), & если \quad z \in Dom(X) \quad и \quad z \notin Dom(Y) \\ f_Y(z) * \sum\limits_{\substack{x<z_i \\ x \in Dom(X)}} f_X(x), & если \quad z \in Dom(Y) \quad и \quad z \notin Dom(X) \\ f_X(z) * \sum\limits_{\substack{y \le z_i \\ y \in Dom(Y)}} f_Y(y) + f_Y(z) * \sum\limits_{\substack{x<z, \\ x \in Dom(X)}} f_X(x), & z \in Dom(X) \quad и \quad z \in Dom(Y) \end{cases}$$

Рассмотрим пример:

$$X: Dom(X) = \{1,3,5\},\ f_X(1) = 0.1,\ f_X(3) = 0.2,\ f_X(5) = 0.7$$

$$Y: Dom(Y) = \{2,4,6\},\ f_Y(2) = 0.3,\ f_Y(4) = 0.6,\ f_Y(6) = 0.1$$

$$Z = \max(X,Y):$$

$$Dom(Z) = \{1,2,3,4,5,6\}$$
$$f_Z(1) = 0.1 \cdot 0 = 0,$$
$$f_Z(2) = 0.3 \cdot 0.1 = 0.03,$$
$$f_Z(3) = 0.2 \cdot 0.3 = 0.06,$$
$$f_Z(4) = 0.6 \cdot (0.1 + 0.2) = 0.18,$$
$$f_Z(5) = 0.7 \cdot (0.3 + 0.6) = 0.63$$
$$f_Z(6) = 0.1 \cdot (0.1 + 0.2 + 0.7) = 0.1$$

Расчет функции плотности вероятности случайной величины Z начинается с формирования отсортированного массива элементов $Dom(X) \bigcup Dom(Y)$. Затем за один проход по отсортированному массиву считается функция плотности. Сложность расчета, таким образом, получается O((m + n)log(m + n)).

Аналогично вычисляется операция минимума: Z = min(X,Y).

$$Dom(Z) = Dom(X) \bigcup Dom(Y)$$

$$f_Z(z_i) = \begin{cases} f_X(z) * \sum\limits_{\substack{y>z_i \\ y \in Dom(Y)}} f_Y(y), & если \quad z \in Dom(X) \quad и \quad z \notin Dom(Y) \\ f_Y(z) * \sum\limits_{\substack{x>z_i \\ x \in Dom(X)}} f_X(x), & если \quad z \in Dom(Y) \quad и \quad z \notin Dom(X) \\ f_X(z) * \sum\limits_{\substack{y \ge z_i \\ y \in Dom(Y)}} f_Y(y) + f_Y(z) * \sum\limits_{\substack{x>z, \\ x \in Dom(X)}} f_X(x), & z \in Dom(X) \quad и \quad z \in Dom(Y) \end{cases}$$

Сложность расчета так же как для операции минимума $O(m+n\cdot)\log(m+n))$.

Введем операцию исключающего выбора одной величины из множества $\{X_i\}, i=1,\ldots,n$ с вероятностями $p_i, i=1,\ldots,n,$ где p_i – вероятность того, что будет выбран величина X_i. Эта операция обозначается далее как $\coprod_{i=1}^{n}(X_i, p_i)$.

Пусть $Z=\coprod_{i=1}^{n}(X_i, p_i)$.

Тогда

$$Dom(Z)=\bigcup_{1\le i\le k} Dom(X_i)$$

$$f_Z(Z=z)=\sum_{z\in Dom(X_j)} p_j\cdot f_{Xj}(z), z\in Dom(Z)$$

.

Последняя формула есть по сути формула полной вероятности.

Расчет функции плотности Z начинается с формирования отсортированного массива элементов $\bigcup_{1\le i\le k} Dom(X_i)$.

Затем за один проход по этому массиву считается функция плотности. Сложность расчета, таким образом,

$$(\sum_{i=1}^{n} n_i)\log(\sum_{i=1}^{n} n_i), \quad \textit{где } n_i \text{ - мощность множества } Dom\ (X_i).$$

Теперь можно рассчитать плотность распределения параметров QoS для составных Web-сервисов. Различные языки конструирования сервисов предлагают разные конструкции для управления ходом выполнения атомарных сервисов.

Рассмотрим управляющие конструкции в составном Web-сервисе. Существует 5 базовых структурных конструкций [2], каждая из которых организует сервисы-компоненты уникальным образом.

1. Последовательность – Sequence(a_1, a_2, a_3, …, a_n): множество сервисов (a_1, ..., a_n) выполняется последовательно.
2. Параллельность – Parallel(a_1, a_2, a_3, …, a_n): несколько сервисов (a_1, ..., a_n) выполняются одновременно и синхронизировано объединяются результаты их выполнения.
3. Исключающий выбор – Choice(a_1, a_2, a_3, …, a_n): из нескольких сервисов (a_1, ..., a_n) выполняется только один.
4. Дискриминатор – Discriminator(a_1, a_2, a_3, …, a_n): несколько сервисов (a_1, ..., a_n) выполняются одновременно, но в результате не синхронизируются, т.е. первый выполнившийся Web-сервис отмечает выполнение всей конструкции и остальные сервисы игнорируются.
5. Цикл – Loop(a): атомарный или составной WS, контролируемый условием LC: сервис выполняется итеративно пока условие выполнено.

Различные стандарты и продукты для составления Web-сервисов включают другие конструкции в дополнение к указанным выше. Можно выделить еще несколько конструкций [2, 3]:

- Чередующаяся параллельная обработка: набор сервисов (a_1, a_2, a_3, …, a_n) выполняется последовательно, но в произвольном порядке.
- Множественный выбор: из набора сервисов (a_1, a_2, a_3, …, a_n) некоторое подмножество ожжет выполняться одновременно.
- Слияние из m по n: используется вместе с конструкцией параллельного выполнения или множественного выбора и после выполнения n параллельных сервисов следующий сервис может быть вызван только тогда, когда выполнены определенные m сервисов.

Эти конструкции покрывают большинство методов управления Web-сервисами, которые предлагают ведущие продукты в этой области, такие как BPEL и OWL-S.

Последние три дополнительные управляющие конструкции могут быть эквивалентно (в смысле вычисления показателей QoS) сведены к основным конструкциям, что будет показано далее.

Определим, как посчитать значения параметров QoS для каждой из этих конструкций. Предположим, что величины QoS атомарных сервисов независимы, т. е., например, время выполнения одного сервиса не влияет на время выполнения следующего в последовательности.

Для каждого Web-сервиса a считаем заданными 3 параметра QoS, а именно, время отклика, стоимость и надежность, обозначенные T(a), C(a) и R(a) соответственно. Для составного сервиса, обозначенного как w, показатели QoS могут быть получены из показателей качества составляющих его сервисов с помощью операций, рассмотренных в разделе. В данной работе рассматривается только нормальное выполнение Web-сервисов и не учитываются исключительные ситуации, когда происходят ошибки. Ошибки и исключения в составных сервисах, а также вопросы транзакций подробно рассматриваются, например, в [5]. Расчет показателей QoS в присутствии ошибок требует хорошо проработанной модели обработки ошибок, которая сильно зависит от реализации составного сервиса. Поэтому этот вопрос оставлен за пределами данной работы.

Последовательность: w состоит из последовательности сервисов (a_1, a_2, a_3, …, a_n).

Стоимость w равняется сумме стоимостей составляющих его Web-сервисов:

$$C(W) = \sum_{i=1}^{n} C(a_i)$$

Время отклика w является суммой времени откликов всех его составляющих:

$$T(W) = \sum_{i=1}^{n} T(a_i)$$

Надежность w – произведение надежностей составляющих его сервисов:

$$R(W) = \prod_{i=1}^{n} R(a_i)$$

Параллельность: w состоит из сервисов (a_1, a_2, a_3, …, a_n), выполняющихся одновременно с последующим объединением результатов их выполнения.

Стоимость w равняется сумме стоимостей составляющих его Web-сервисов:

$$C(W) = \sum_{i=1}^{n} C(a_i)$$

Время отклика w является максимумом из множества времен откликов всех его составляющих:

$$T(W) = \max_{1 \le i \le n} \{T(a_i)\}$$

Надежность w – произведение надежностей составляющих его сервисов:

$$R(W) = \prod_{i=1}^{n} R(a_i)$$

Исключающий выбор: w состоит из множества исключающих друг друга сервисов (a_1, a_2, a_3, …, a_n), каждый из которых связан с вероятностью p_i (для a_i), показывающей вероятность того, что выполнится a_i.

Стоимость w – исключающий выбор из стоимостей составляющих его сервисов с вероятностями выбора данного сервиса:

$$C(W) = \coprod_{i=1}^{n} (C(a_i), p_i)$$

Время отклика – исключающий выбор из времен откликов составляющих его сервисов с вероятностями выбора данного сервиса:

$$T(W) = \coprod_{i=1}^{n} (T(a_i), p_i)$$

Надежность – исключающий выбор из надежностей составляющих:

$$R(W) = \coprod_{i=1}^{n} (R(a_i), p_i)$$

Дискриминатор: w состоит из параллельно выполняющихся сервисов (a_1, a_2, a_3, …, a_n) без синхронизации.

Стоимость w равняется сумме стоимостей составляющих его Web-сервисов:

$$C(W) = \sum_{i=1}^{n} C(a_i)$$

Время отклика является минимумом времен отклика составляющих, т. к. первый выполненный компонент отмечает всю конструкцию w выполненной:

$$T(W) = \min_{1 \le i \le n} \{T(a_i)\}$$

Сервис w считается не выполненным только если все его составляющие Web-сервисы не выполнятся. Следовательно, надежность w:

$$R(W) = 1 - \coprod_{i=1}^{n} (1 - R(a_i))$$

где: $1\text{-}R(a_i)$ обозначает обратную величину к заданной $R(a_i)$.

В цикле сервис запускается несколько раз в зависимости от условия LC. В [3] предполагается, что число вызовов цикла подчиняется геометрическому распределению. Но

геометрическое распределение не учитывает, то явление, что с увеличением числа итераций выполнения сервиса увеличивается вероятность выхода из цикла. В [6] считается, что число итераций подчиненным нормальному распределению, что опять-таки не всегда соответствует действительности. Вместо того, чтобы считать распределение заданным, можно напрямую указывать дискретную случайную величину для числа итераций, определяя ее область определения и плотность распределения. Для цикла стоимость, время отклика и надежность считаются по формулам:

$$C(W) = \prod_{i=1}^{n} \left(C_a(l), f_{L(a)}(l)\right),$$

$$R(W) = \prod_{i=1}^{n} \left(R_a(l), f_{L(a)}(l)\right),$$

$$T(W) = \prod_{i=1}^{n} \left(T_a(l), f_{L(a)}(l)\right),$$

где: $f_{L(a)}(l), 1 \le l \le c$ функция плотности числа итераций в цикле и $T_a(l) = \sum_{i=1}^{l} T(a_i), \quad C_a(l) = \sum_{i=1}^{l} C(a_i), \quad R_a(l) = \sum_{i=1}^{l} R(a_i).$

Рассмотрим прочие конструкции.

При чередующейся параллельной обработке, которая поддерживается BPEL и OWL-S, может быть просто рассмотрена с точки зрения расчета показателей QoS как последовательность. Следовательно, для этой конструкции получаются те же расчетные формулы, что и для последовательности.

Множественный выбор, поддерживающийся BPEL и OWL-S, может рассматриваться как конструкция параллельности с некоторыми модификациями функций плотности для показателей QoS. Пусть w – составной сервис, представляющий множественный выбор из сервисов (a_1, a_2, a_3, …, a_n), каждый из которых выбирается с вероятностью a_i. Очевидно, когда сервис не вызывается, его стоимость и время отклика равны 0 и надежность равна 1. Таким образом, области определения для новых стоимости *C'* и времени *T'* расширяются:

$$Dom(C'(a_{i})) = Dom(C(a_i)) \bigcup \{0\}$$

$$Dom(T'(a_{i})) = Dom(T(a_i)) \bigcup \{0\}.$$

Функции плотности меняются:

$$f_{C'(a_i)}(c) = p_i \cdot f_{C(a_i)}(c), если \quad c \in Dom(C(a_i))$$

$$f_{C'(a_i)}(c) = 1 - p_i, если \quad c = 0$$

$$f_{T'(a_i)}(t) = p_i \cdot f_{T(a_i)}(t), если \quad t \in Dom(T(a_i))$$

$$f_{T'(a_i)}(t) = 1 - p_i, если \quad t = 0$$

Функция плотности для надежности изменится:

$$f_{R'(a_i)}(0) = p_i \cdot f_{R(a_i)}(0), \; f_{R'(a_i)}(1) = 1 - p_i + p_i \cdot f_{R(a_i)}.$$

Стоимость, время отклика и надежность w, таким образом, рассчитываются так же как для параллельности.

Наконец, слияние из m по n, поддерживаемое OWL-S, может быть сведено к параллельности. Стоимость слияние из m по n считается также как для конструкции параллельности. Для расчета времени отклика должна использоваться новая функция Min-m( ), которая возвращает m-тую наименьшую величину из множества случайных величин. Min-m( ) является более общей функцией для минимума, который описан выше, и ее формула поэтому в данной работе опускается.

Показатели QoS для Web-сервисов, объединяющих структурные конструкции, могут быть рекурсивно получены по приведенным выше формулам. Составные сервисы, использующие другие не структурные конструкции, не подходят для расчетов показателей QoS по составляющим его частям. Одной из таких конструкций является связь(link), реализованная в BPEL. Связь – конструкция, созданная для синхронизации двух Web-сервисов, выполняющихся внутри любой другой конструкции (кроме цикла) на любом уровне вложенности. Для такой конструкции практически нереально предсказать выполнение.

Расчет качества по приведенной в предыдущем разделе схеме можно производить рекурсивно, как показано в [11].

При расчете полученной схемы растет область определения результирующей случайной величины. Например, при сложении k случайных величин, имеющих по n элементов в области определения, полученная случайная величина будет иметь в худшем случае порядка n^k элементов в области определения. Чтобы уменьшить это число после каждой операции до приемлемого размера можно группировать элементы. Если несколько последовательных элементов в области определения функции имеют одно и то же значение, они заменяются одним значением и рассчитывается групповая вероятность. Для этого вводится группирующая величина. Подробно расчет результирующей случайной величины приведен в [12].

Задача поиска оптимальной группирующей случайной величины свелась к рассмотрению двух алгоритмов: рекурсивного и «жадного». Как и предполагалось, при увеличении размерности задачи время работы намного быстрее возрастает для рекурсивного алгоритма, чем для «жадного». При этом разница в точности не велика, поэтому далее для тестирования модели качества использовался «жадный» алгоритм как более гибкий.

Для проверки модели будем использовать только время отклика T( ), т. к. его можно легко измерить в действительности. Для сравнения с рассчитанным по модели временем используется реально измеренное время отклика составного сервиса.

На рисунке 1 представлена типичная модель обработки заказа для он-лайн магазина. На самом высоком уровне в составном сервисе находится конструкция последовательности, которая состоит из сервисов OrderCreation, StockPaymentCheck и ShippingArrangement. Сервис tockPaymentCheck представляет собой параллельное выполнение сервисов PaymentCheck и StockCheck. PaymentCheck в свою очередь является конструкцией выбора, состоящей из сервисов PaypalValidation и CreditCardValidation, выбор между которыми определяется пользователем через запрос. StockCheck – это цикл вызовов сервиса GetItem, который выполняется для каждой записи в заказе. Если требуемое есть на складе, доставка заказывается оттуда (GetFromWarehouse), иначе отправляется заказ на доставку поставщику (GetFfromVendor).

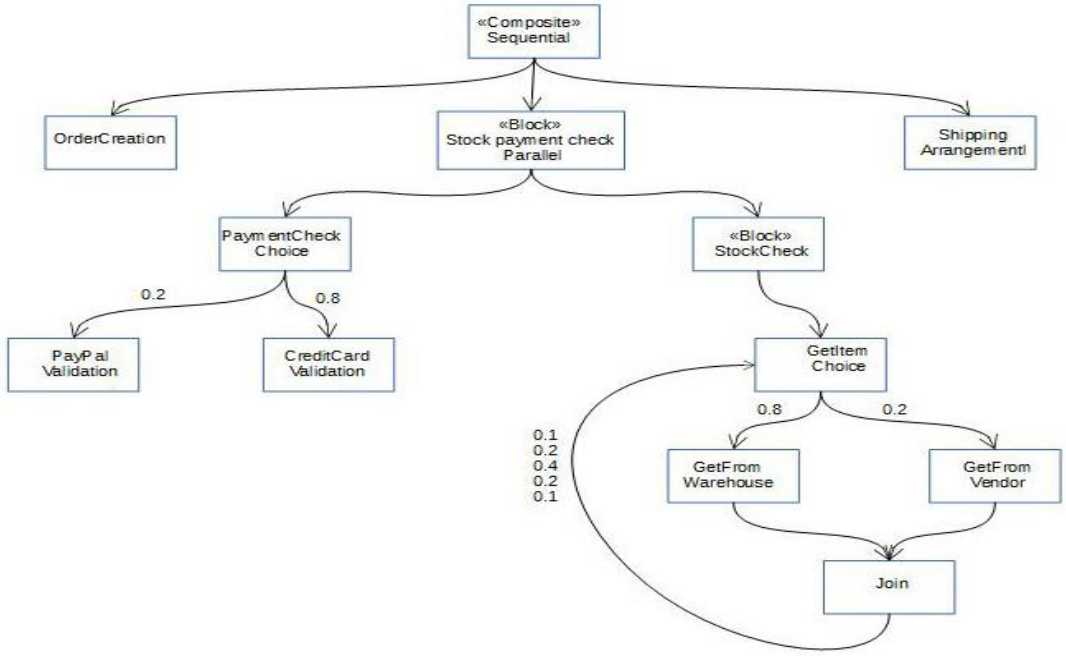

Рисунок 1. *Модель обработки заказа – тестовый сервис (разработано автором)*

Каждый элементарный сервис в составе композитного сервиса не делает ничего, кроме случайной задержки, выбираемой из массива, сгенерированного для каждого сервиса и загруженного в память заранее до начала эксперимента. Массивы формируются по определенным законам распределения, параметры которых приведены в [11]. Кроме того, вероятности выбора того или иного сервиса фиксированы и указаны на рисунке 1.

Приведем результаты моделирования, полученные в [13]. Гистограмма времени отклика строится на основе нескольких серий вызовов составного сервиса. Потом это же время считается по приведенным формулам с применением «жадного» алгоритма сокращения области определения (т. к. было показано [13], что этот алгоритм работает оптимально). Получено следующее:

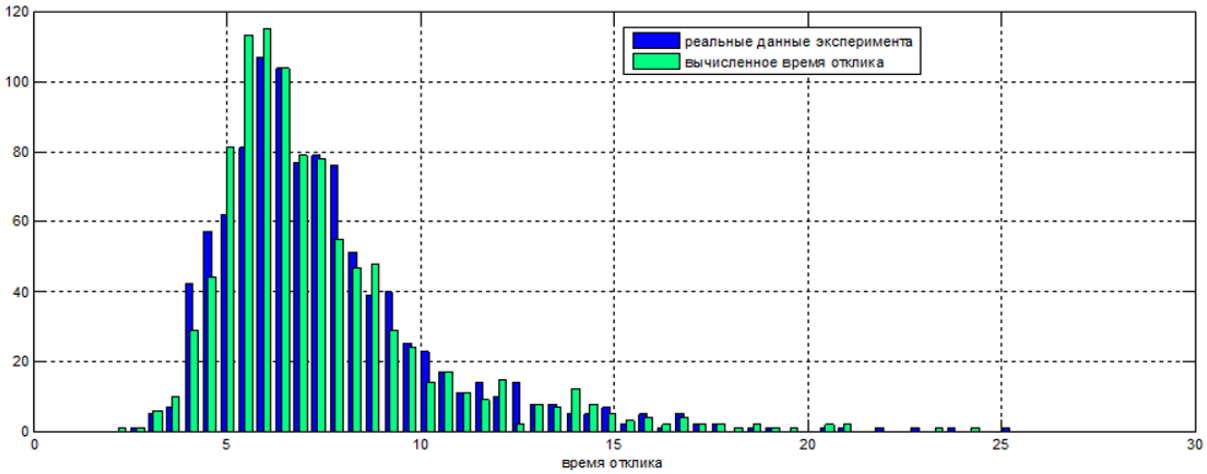

Рисунок 2. *Гистограмма времени отклика –*
реальные данные и вычисленное время (разработано автором)

При расчете без сокращения области определения (рис. 3) на каждом шаге расчета время работы алгоритма существенно возрастает, а изменение отклонения от практически измеренного результата не заметно:

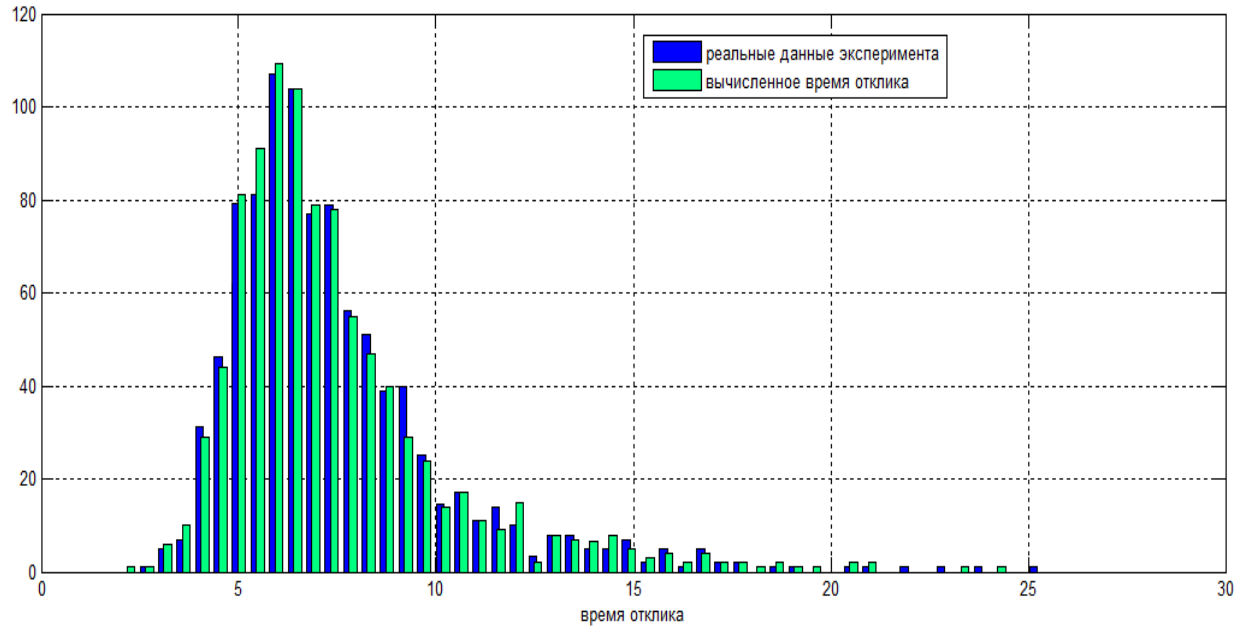

Рисунок 3. *Время отклика (зеленым) без сокращения области определения (разработано автором)*

Далее на графике (рис. 3) показано отклонение практически полученной функции плотности для времени отклика (показана на гистограмме синим) и посчитанной по схеме предлагаемой модели без применения алгоритмов сокращения области определения (на гистограмме, показанной зеленым).

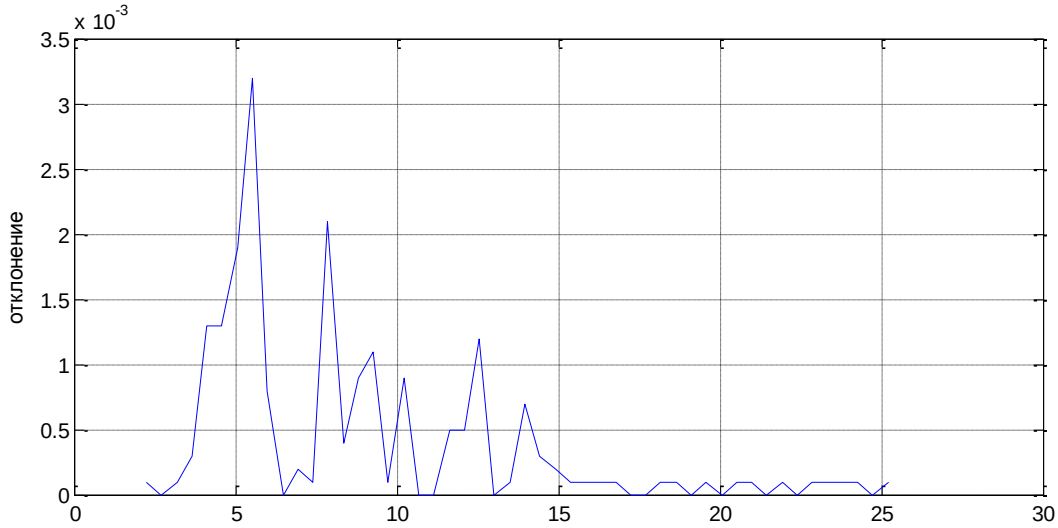

Рисунок 4. *Разница в точности (разработано автором)*

Далее на графике (рис. 4) показано отклонение практически полученной функции плотности для времени отклика и посчитанной по схеме предлагаемой модели без применения алгоритмов сокращения области определения.

Рассмотренные в данной работе вопросы качества Web-сервисов поднимают одну из важных проблем объединения отдельных сервисов в потоки. Такое объединение предполагает, что приложение может оптимально выбрать поставщиков отдельных сервисов в соответствие с требованиями QoS. В вероятностной среде оценка QoS составного сервиса не всегда может напрямую вычисляться по оценкам QoS его составных частей. Предложенная модель оценки качества сервисов учитывает особенности каждого атомарного Web-сервиса. Эта модель

рассматривает также 5 базовых структурных конструкций, каждая из которых организует сервисы-компоненты уникальным образом. Оценка качества композитного Web-сервиса с помощью предложенной методики позволяет гарантировать, что сервис сработает на определенном уровне качества (QoS), что доказано проведенными вычислительными экспериментами, результаты которых представлены в работе.

ЛИТЕРАТУРА

1. Cardoso J., Sheth A., Miller J., Arnold J., Kochut K. Quality of service for workflows and web service processes // Journal of Web Semantics, 2004. Vol. 1. P. 281-308.
2. El Haddad, J., M. Manouvrier, and M. Rukoz, TQoS: Transactional and QoS-Aware Selection Algorithm for Automatic Web Service Composition. IEEE Transactions on Services Computing, 2010. 3(1): p. 73-85.
3. Gillmann M. et al. Workflow management with service quality guarantees // Information Sciences, 2002. Vol.1 P. 228-239.
4. Hwang S.-Y. et al. A probabilistic approach to modeling and estimating the QoS of web-services-based workflows // Information Sciences, 2007. Vol. 177. P. 5484-5503
5. Hwang S.-Y., Tang J. Consulting past exceptions to facilitate workflow exception handling // Decision Support Systems (DSS), 2004 Vol. 37(1). P. 49-69.
6. Jaeger M. C., Rojec-Goldmann G., Muehl G. QoS aggregation for Web service composition using workflow patterns // Proceedings of IEEE 8th International Conference on Enterprise Distributed Object Computing (EDOC04), 2004. P. 149-159.
7. Keen M., Cavell J., Hill S., Kee C., Neave W., Rumph B., Tra H. BPEL4WS Business Processes with WebSphere Business Integration: Understanding, Modeling, Migrating. – International Business Machines Corporation. 2004. – 363 p.
8. Lin C.-F., Sheu R.-K., Chang Y.-S., Yuan S.-M. A relaxable service selection algorithm for QoS-based web service composition // Information and Software Technology. 2011; 53(12):1370-1381.
9. Wang H.-C., Lee C.-S., Ho T.-H. Combining subjective and objective QoS factors for personalized Web service selection // Expert Systems with Applications, 2007. Vol. 2. P. 571-584.
10. Zhao L., Ren Y., Li M., Sakurai K. Flexible service selection with user-specific QoS support in service-oriented architecture // Journal of Network and Computer Applications. 2012;35(3):962-973.
11. Волушкова В. Л. Школенко Е. Ю. Модель оценки качества Web-сервисов // Вестник Московского университета имени С. Ю. Витте. Материалы XLI международной конференции «Информационные технологии в науке, образовании, телекоммуникации и бизнесе IT+SE`2013», 2013. – с. 213-216.
12. Волушкова В. Л. Оценка качества Web-сервисов // ИТНОУ: информационные технологии в науке, образовании и управлении.2017. № 2(2), стр. 12-17.
13. Волушкова В. Л. Школенко Е. Ю. Алгоритмическая реализация модели качества web-сервисов // Интернет-журнал «Науковедение». 2014 №1 (20) [Электронный ресурс] – М.: Науковедение, 2014 – Режим доступа: https://elibrary.ru/download/elibrary_21541388_15669991.pdf, свободный. – Загл. с экрана. – Яз. рус., англ.
14. Кузнецов С. Как поймать в Сети злоумышленника? // Открытые системы, 2, 2007, стр. 102-109.
15. Ньюкомер Э. Веб-сервисы. – СПб.: Питер, 2003. – 256 с.

Volushkova Vera L'vovna
Tver state university, Russia, Tver
E-mail: w2l@pisem.net

Modeling the quality of web-services

Abstract. A Web service is a programming interface that describes a set of operations that can be remotely called over the network through standardized XML messages. A Web service is a unit of modularity when using a service-oriented application architecture.

For service providers, Quality of Service (QoS) is an important aspect of Web services technology. Services without guaranteed quality can be acceptable in simple maintenance cases. QoS is a critical factor when choosing Web services as components for complex business processes.

In this paper, the quality of Web services is modeled. The created model of quality allows not only to take into account the peculiarities of building composite services, but also the uncertainty in the work of services connected with the Web-environment. For this, the QoS indicators of both atomic and composite Web services are considered probabilistic.In a probabilistic environment, the QoS estimation of a composite service can not always be directly calculated by estimating the QoS of its constituent parts. The model considers five basic structural designs, each of which organizes the service components in a unique way.

Using the proposed methodology, you can evaluate the quality of the composite Web service and ensure that the service will work at a certain level of quality. This is proved by the computational experiments carried out, the results of which are presented in the paper.

Keywords: web-service; composite web-service; quality of web-service; quality of service attribute; response time; reliability; optimization; recursive algorithm; "greedy" algorithm; Web service composition