УДК 681.321

А.В. БОЯРЧУК

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

РАЗРАБОТКА МАРКОВСКОГО ГРАФА СОСТОЯНИЙ КОМПОЗИТНОЙ СЕРВИС-ОРИЕНТИРОВАННОЙ АРХИТЕКТУРЫ

Проведен анализ функциональной модели системы сервис-ориентированой архитектуры. Разработаны графы состояний системы в различных условиях. Выполнена нормализация графа для построения марковской модели системы.

SOA-системы, функциональная модель, марковский граф

Введение

Постановка задачи. Использование технологий web-сервисов – серьезная тенденция в области разработки современных распределенных информационных систем. Системы сервис-ориентированной архитектуры в настоящее время широко применяются при разработке бизнес-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 качества в обслуживании такого рода систем являются актуальными [1].

Архитектура web-сервисов создает множество проблем, в большинстве своем зависящих от специфики функционирования самих сервисов. Применяемые экспериментальные подходы оценивания показателей сервис-ориентированных систем и известные аналитические модели не дают возможности исследовать поведение таких систем в условиях внешних воздействий при динамически изменяющихся параметрах [2].

Целью данной статьи является обоснование возможности и разработка графа состояний марковской модели функционирования композитной системы сервис-ориентированной архитектуры (КСОА).

Анализ функциональной модели КСОА

Развитие web-технологий привело к появлению функционально новой архитектуры web-приложений, которое стало известно как композитные службы [3, 4]. В данном случае под композитностью службы следует считать такой тип архитектуры системы, которая представлена многоуровневой иерархией с регулярной и рекурсивно-вложенной структурой. Приходящие запросы система декомпозирует на каждом «горизонтальном» уровне своей архитектуры таким образом, что каждый из находящихся на данном уровне композитов получает предназначенную для него часть запроса и, в свою очередь, аналогичным образом разделяет запрос дальше и передает его на уровень ниже. Достигнув таким образом самого нижнего уровня композитной системы, исходный запрос уже разделен на атомарные операции, которые выполняются на нижнем уровне как единый блок.

Структурный подход базируется на разработке композитного web-сервиса как системы с многоуровневой иерархией с регулярной и рекурсивно-вложенной структурой. В общем случае композитные web-сервисы на нижнем уровне иерархии – заранее разработанные сервисы с известными характеристиками (готовность, безопасность и т.д.). Показатели композитного web-сервиса могут быть обеспечены следующими способами:

– разработкой технологий компонентных и композитных web-сервисов с требуемой функциональностью и надежностью;

– разработкой специальных процедур поддержки отказоустойчивости с помощью базовых элементов композитных web-сервисов;

– применением различных типов избыточности и диверсности на различных уровнях иерархии композитных web-сервисов.

Частным случаем реализации «вертикальной» композитной службы может служить программный комплекс туристического агентства, реализующего сервисы бронирования отелей, заказа авиационных билетов, аренды транспортного средства.

Для выполнения своих функций веб-система турагенства взаимодействует с открытым интерфесом системы бронирования авиабилетов (AirFrance/KLM, British Airways, …), системами бронирования отелей (Sofitel, Holiday Inn, …) и сервисами бронирования автомобилей (Hertz, RentACar, …). 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.

Построение графа состояний системы

Моделью принято считать абстрактный формализованный образ произвольной системы, имеющий с ней необходимую степень сходства. На основании проведенного анализа будут сделаны выводы об особенностях функционирования системы в зависимости от типов воздействий.

Исходя из логики функционирования проектируемой системы, по мнению автора, ее следует позиционировать как систему с неограниченным ожиданием и неограниченным числом заявок. Для первого фактора (неограниченное ожидание) принято допущение, касающееся того, что для моделирования устанавливается бесконечно большая очередь поступающих пользовательских запросов; второй (неограниченное число заявок) соответствует реальному поведению web-систем. Кроме того, в качестве допущения принят тот факт, что моделируемая система – это система без дообслуживания, т.е. пользовательские запросы, частично не удовлетворенные системой по каким-либо причинам и не принятые пользователем, отбраковываются системой и не подлежат повторной обработке. В этом разделе построена и обоснована первая, самая простая, модель системы без учета воздействий. Под воздействиями в данном случае понимается все возможные аномалии в работе системы, не поддающееся контролю со стороны самой системы. Детально таксономия воздействий будет представлена в соответствующем разделе.

По мнению автора, следующие состояния наиболее точно отражают поведение системы в случае ее «идеального» функционирования, т.е. без воздействий.

Проанализируем подробно каждое из состояний:

1. *Инициализация*. Первоначальное состояние системы, в которое она переходит после включения. Во время инициализации происходит: а) старт всех модулей системы; б) установление соединения с БД; в) формирование первичного множества целевых web-сервисов; г) конфигурирование системы и перевод ее в один из режимов – максимальной надежности, максимальной реактивности или минимальной загрузки; д) инициализация очереди клиентских запросов.

2. *Ожидание запроса*. Второе состояние системы после *инициализации*. Все модули системы запущены, целевые web-сервисы уже зарегистрированы в системе. Очередь заявок пуста и ожидает поступления пользовательских запросов.

3. *Получение запроса из очереди*. В очередь начинают *поступать* первые запросы, система поочередно выбирает их и отправляет на декомпозицию.

4. *Декомпозиция запросов*. Выбранные из очереди запросы поступают на вход блока, занимающегося *декомпозицией* запросов на верхнем уровне. Определяется категория запроса, т.е. количество композитов в нем (для туристического агентства – подзапросы бронирования авиабилетов, бронирования отеля, аренды автомобиля); из множества зарегистрированных целевых web-сервисов выбираются те, которые могут удовлетворить полученные подзапросы.

5. *Обработка запросов.* На этом этапе система *непосредственно* выполняет обработку запросов с помощью целевых web-сервисов. В соответствии с рассмотренной ранее структурой композитного сервиса приходящие запросы декомпозируются на каждом уровне «вложенности» композитных серверов, после чего передаются на обработку к целевым сервисам. Дальнейшее поведение системы (без учета воздействий) зависит от того, устраивает ли пользователя результат, полученный от целевых сервисов, или нет. В случае принятия пользователем предложенного результата, система формирует ответ и отсылает пользователю. В противном случае полученные результаты отбраковываются, и пользователю возвращается сообщение о невозможности удовлетворения его запроса. Граф состояний системы без учета воздействий представлен на рис. 1.

На следующем этапе построения модели проектируемой системы учтены состояния, появляющиеся в случае отсутствия информационного ресурса для удовлетворения пользовательских заявок.

Возвращаясь к примеру туристического агентства, информационным ресурсом являются данные о доступности авиабилетов, сведения о свободных номерах в отеле, а также наличие автомобиля требуемой марки на указанные сроки. Отсутствие информационного ресурса в этом случае может быть обусловлено: а) недоступностью целевого web-сервиса, который отвечает за предоставление данной информации; б) отсутствием информации у web-сервиса, удовлетворяющей заданным критериям пользовательского запроса.

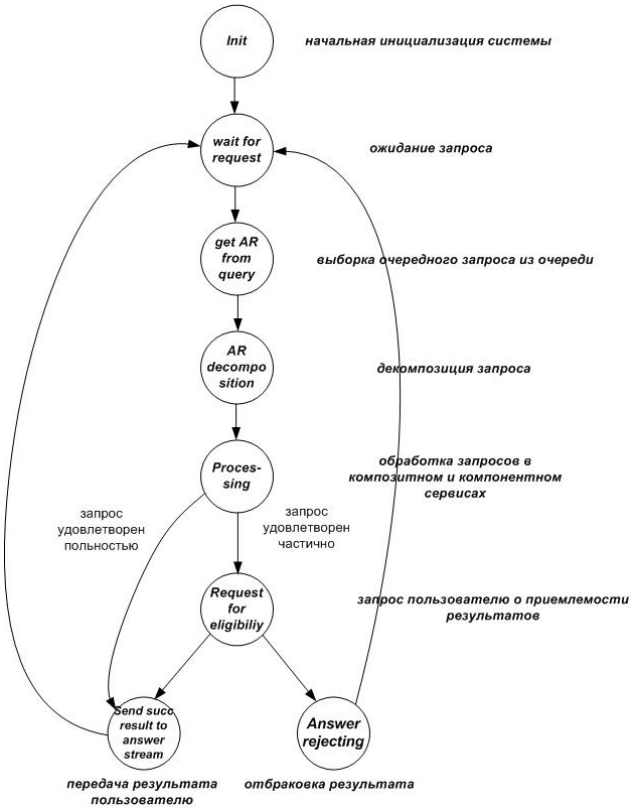

Рис. 1. Модель системы без учета воздействий

Для функционирования такой модели необходимо ввести состояние обработки исключения на верхнем уровне. Под обработкой исключительной ситуации понимается попытка обратиться за требуемой информацией к альтернативному сервису, которые может предоставить аналогичную информацию, а также попытка выполнить неудавшийся запрос с близкими параметрами.

Модель системы с учетом отсутствия информационного ресурса представлена на рис. 2.

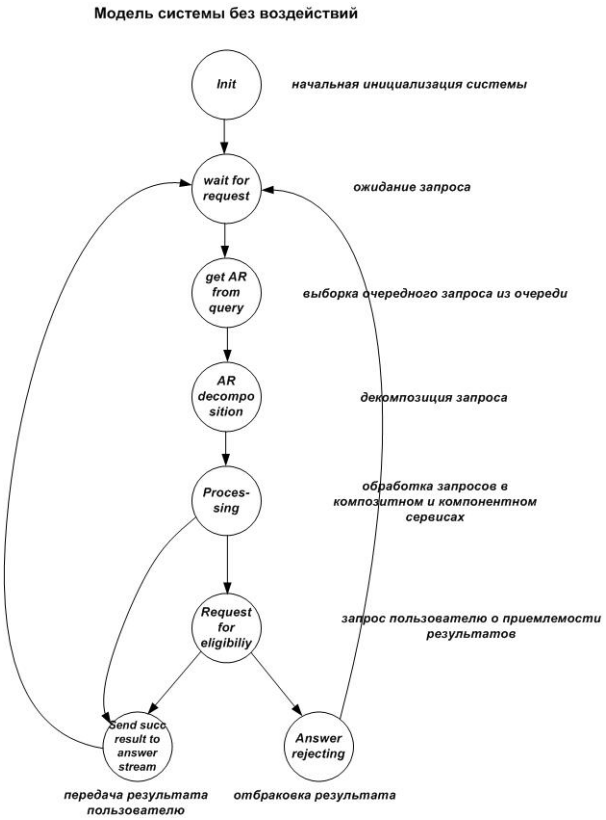

Рис. 2. Модель системы с учетом отсутствия информационного ресурса

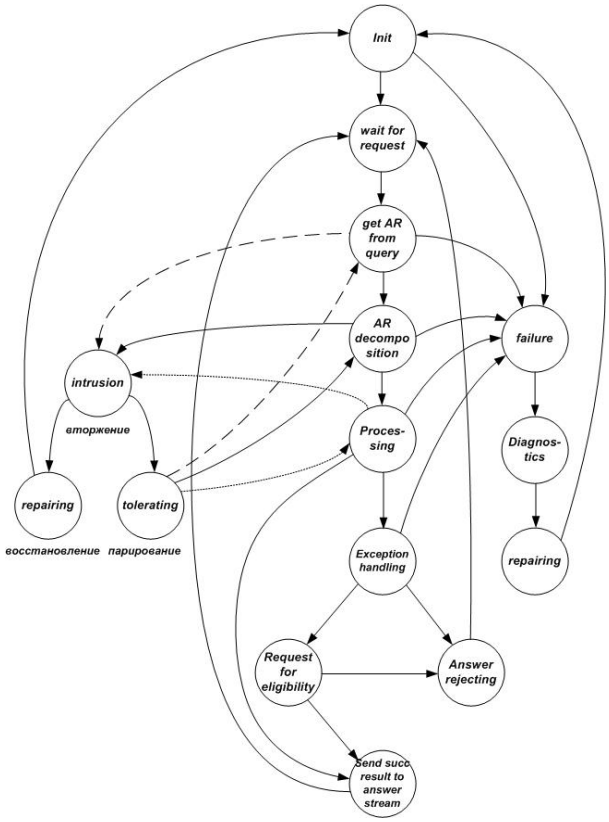

Рис. 3. Граф состояний системы с учетом воздействий

При разработке этого графа состояний системы с учетом воздействий (рис. 3) приняты следующие допущения:

1. В случае *возникновения неисправности* программных или аппаратных модулей системы происходит частичный или полный отказ системы или ее компонентов, диагностирование причины отказа, восстановление системы и ее инициализация.

2. В случае *парированной атаки* система продолжает функционирование, сведения об отраженной атаке протоколируются в журнале безопасности.

3. В случае возникновения *атаки*, которую *невозможно отразить*, возможны следующие варианты поведения системы: а) теряет часть своей функциональности; б) выполняет свои функции в прежнем объеме, но значительно медленнее, в) прекращает функционировать до полного восстановления и повторной инициализации.

Переход к марковской модели

Последняя разработанная модель (рисунок 3) описывает все возможные состояния системы. Однако для этого графа состояний не выполняется условия марковости. Как известно, случайный процесс, протекающий в системе, называется марковским, если он обладает следующим свойством: для каждого момента времени вероятность любого состояния системы в будущем зависит только от ее состояния в настоящем и не зависит от того, когда и каким образом система пришла в это состояние [5].

Таким образом, наша система может быть представлена марковским процессом с дискретными состояниями и непрерывным временем только при условии декомпозиции состояний. Представим систему в виде графа состояний, где вершины – состояния системы, а ориентированные дуги – переходы из состояния в состояние.

Из рис. 3 можно заметить, что условия марковости не выполняются для некоторых состояний: а) переход из состояния обработки запроса к обработке

исключения осуществляется по двум условиям (частичное выполнение запроса и полное невыполнение запроса) и б) для пары состояний «вторжение – парирование» характерно возвращение системы именно в то исходное состояние, из которого система перешла в состояние «вторжение». Чтобы перейти от имеющегося графа к такому, для которого будут выполняться условия марковости, необходимо провести декомпозицию упомянутых состояний, что приведет к незначительному увеличению количества состояний системы. Модель системы после декомпозиции состояний показана на рис. 4.

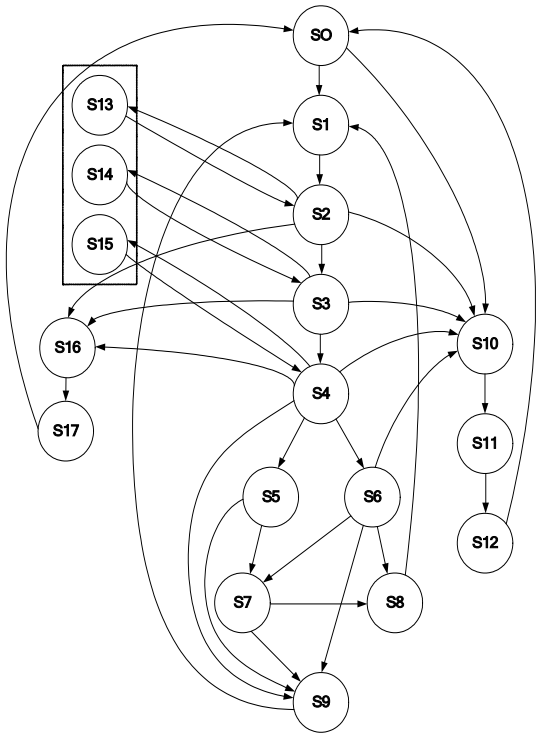

Рис. 4. Неразмеченный граф марковской модели системы

В итоге получаем модель с 18 состояниями: S0 – инициализация системы; S1 – ожидание заявки; S2 – получение заявки из очереди; S3 – декомпозиция заявки; S4 – обслуживание заявки; S5 – обработка исключения для частично обслуженной заявки; S6 – обработка исключения для полностью необслуженной заявки; S7 – запрос пользователю относительно принятия частично обслуженной заявки; S8 – удаление полностью необслуженной заявки; S9 – отсылка пользователю результата обслуживания заявки; S10 – отказ системы по причине неисправности программного / аппаратного обеспечения; S11 – диагностирование неисправности; S12 – восстановление системы; S13 – парирование вторжения, произошедшего в состоянии S2; S14 – парирование вторжения, произошедшего в состоянии S3; S15 – парирование вторжения, произошедшего в состоянии S4; S16 – отказ системы по причине непарированного вторжения; S17 – восстановление системы после вторжения.

Исходя из данных экспертной оценки [6], интенсивности переходов между состояниями графа были размечены следующим образом:

L01 = S0->S1 = 100 1 / час;

L010= S0->S10= 0,1 1 / час;

L12 = S1->S2 = 10000 1 / час;

L23 = S2->S3 = 10000 1 / час;

L210 = S2->S10 = 0,1 1 / час;

L213 = S2->S13 = 0,1 1 / час;

L216 = S2->S16 = 0,1 1 / час;

L34 = S3->S4 = 1000 1 / час;

L310 = S3->S10 = 0,1 1 / час;

L314 = S3->S14 = 0,1 1 / час;

L316 = S3->S16 = 0,1 1 / час;

L410 = S4->S10 = 0,1 1 / час;

L415 = S4->S15 = 0,01 1 / час;

L416 = S4->S16 = 0,1 1 / час;

L46 = S4->S6 = 10 1 / час;

L49 = S4->S9 = 10000 1 / час;

L57 = S5->S7 = 100 1 / час;

L59 = S5->S9 = 100 1 / час;

L67 = S6->S7 = 100 1 / час;

L68 = S6->S8 = 100 1 / час;

L69 = S6->S9 = 10 1 / час;

L610 = S6->S10 = 0,1 1 / час;

L78 = S7->S8 = 10 1 / час;

L81 = S8->S1 = 10000 1 / час;

L91= S9->S1 = 10000 1 / час;

L79 = S7->S9 = 100 1 / час;

L1011 = S10->S11 = 10 1 / час;

L1112 = S11->S12 = 10 1 / час;

L120 = S12->S0 = 10 1 / час;

L132 = S13->S2 = 0,1 1 / час;

L143 = S14->S3 = 0,1 1 / час;

L154 = S15->S4 = 0,1 1 / час;

L1617 = S16->S17 = 100 1 / час;

L170 = S17->S0 = 100 1 / час.

Размеченный марковский граф показан на рис. 5.

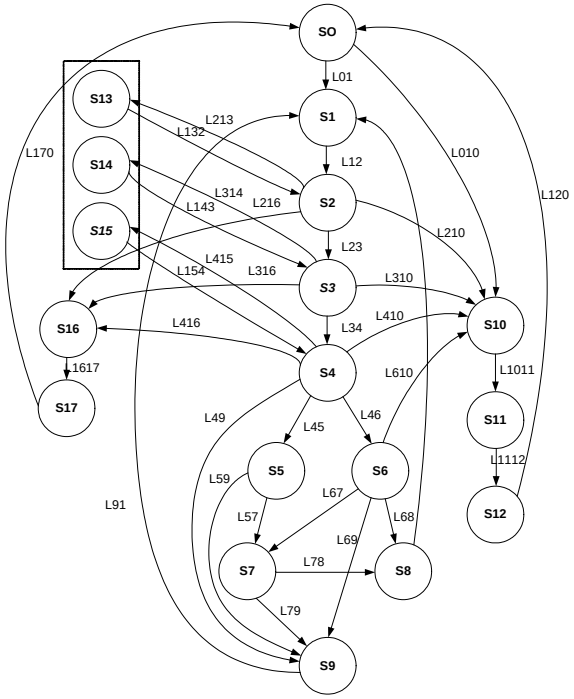

Рис. 5. Размеченный граф марковской модели системы

Заключение

В данной статье были рассмотрены модели функционирования системы сервис-ориентированной архитектуры. Построены графы состояний системы: а) без учета воздействии; б) с учетом отсутствия информационного ресурса; в) с учетом воздействий; г) неразмеченный марковский граф; д) размеченный марковский граф. На основании экспертных данных выполнена разметка полученного марковского графа.

Полученная модель функционирования системы позволяет рассчитать вероятность нахождения системы в каждом из ее состояний при заданных значениях интенсивности переходов.

Литература

1. Менаске Д., Алмейда В. Производительность web-служб. Анализ, оценка и планирование. – СПб: ООО «ДиаСофтЮП», 2003. – 408 с.

2. Tartanoglu F., Issarny V., Romanovsky A., Levy N. Coordinated Forward Error Recovery for Composite Web Services // The 22nd Symposium on Reliable Distributed Systems, Florence, Italy, 2003. – P.167-176.

3. Kharchenko V., Popov P., Romanovsky A., Boyarchuk A., Gorbenko A. CS-TR: 863.

4. Web Services out of Undependable Web Components // School of Computing Science, Univers. of Newcastle, 2004. – P. 36.

5. Kharchenko V., Popov P., Romanovsky A. On Dependability of Composite Web Services with Components Upgraded Online // Proceedings of DSN 2004 Workshop on Architecting Dependable Systems, Italy, 2004. – P. 14-20.

6. Харченко В.С., Лысенко И.В. Теория систем и системный анализ. – Х.: ХАИ, 2003. – С. 130.

7. Боярчук А.В. Моделирование системы управления web-сервисами в условиях различных воздействий // ДЕССЕРТ 2006, Полтава, 2006. – С. 157-162.

Поступила в редакцию 21.01.2008

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