УДК 681.321

А.В. БОЯРЧУК

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

МОДЕЛИРОВАНИЕ СИСТЕМЫ УПРАВЛЕНИЯ ОБНОВЛЕНИЯМИ WEB-СЕРВИСОВ В УСЛОВИЯХ РАЗЛИЧНЫХ ВОЗДЕЙСТВИЙ

Разработана математическая модель системы управления обновлениями web-сервисов как системы массового обслуживания. Исследована надежность системы в условиях различных воздействий.

web-сервисы, гарантоспособные системы, марковский граф, математическая модель, надежность

Введение

Постановка задачи. Для решения проблемы организации производительного, надежного и безопасного взаимодействия сложных корпоративных систем в качестве одного из наиболее эффективных инструментов применяется концепция сервис-ориентированной архитектуры приложений. Проблема онлайн-обновлений систем сервис-ориентированной архитектуры достаточно актуальна [1 – 4]. Основной причиной для построения систем онлайн-обновления web-сервисов является потребность улучшения и/или добавления в них функциональности, а также коррекция возможных ошибок в обновленных версиях web-сервисов.

Применяемые на сегодняшний день методы [4, 6] не решают описанную проблему комплексно, а лишь обеспечивают механизм замещения версий обновленных web-сервисов. **Цель статьи** – исследование надежности приложений сервис-ориентированной архитектуры путем моделирования и оценивания характеристик системы управления обновлениями web-сервисов.

Результаты исследований

Обоснование подхода к разработке модели. 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.

Существует два принципиально различных подхода к моделированию системы. Первый, архитектурный, описывает поведение системы, основываясь на логике работы компонентов системы. Второй, функциональный подход к моделированию системы, описывает ее работу, основываясь на представлении системы как «черного ящика». Преимуществом такой модели является, во-первых, прозрачная схема функционирования системы, и, во-вторых, система получается более удобной для восприятия и анализа за счет сравнительно небольшого количества состояний и переходов между ними. Исходя из логики функционирования исследуемой системы, ее следует рассматривать как систему массового обслуживания с неограниченным ожиданием, неограниченным числом заявок и без дообслуживания. Далее моделируется поведение системы без учета и с учетом различных типов неисправностей и воздействий.

Под воздействиями в данном случае понимаются все возможные события, вызвавшие аномалии в работе системы, не поддающиеся контролю со стороны самой системы. Следующие состояния наиболее полно отражают поведение системы в случае ее функционирования без воздействий: инициализация; ожидание запроса; получение запроса из очереди; декомпозиция запросов; обработка запросов.

При построении модели проектируемой системы должны быть учтены состояния, появляющиеся в случае отсутствия информационного ресурса для удовлетворения пользовательских заявок. Под информационным ресурсом понимаются данные, которые запрашиваются у целевого ресурса исследуемой системой. Отсутствие информационного ресурса в этом случае может быть обусловлено, во-первых, недоступностью целевого web-сервиса, который предоставляет данную информацию; во-вторых, отсутствием информации у web-сервиса, удовлетворяющей заданным критериям пользовательского запроса.

Кроме того, в такой модели необходимо учесть состояние обработки исключения на верхнем уровне. Под обработкой исключительной ситуации понимается попытка обратиться за требуемой информацией к альтернативному сервису, который может предоставить аналогичную информацию.

Таксономия воздействий. Согласно фундаментальной концепции гарантоспособности [1], угрозами для компьютерных систем являются ошибки (errors), неисправности (faults) и отказы (failures). Под ошибкой понимается состояние системы, которое может привести к последующему отказу; отказ происходит, когда ошибка достигает интерфейса системы и видоизменяет системную услугу; неисправность – реальная или гипотетическая причина ошибки. Все неисправности системы могут быть объединены в одну из трех групп: неисправности разработки (design faults), физические неисправности (physical faults) и неисправности взаимодействия (interaction faults).

Неисправности вследствие злого умысла (malicious faults) могут быть, в свою очередь, разделены на неисправности, причиненные разрушительной логикой (malicious logic – троянские кони, вирусы, черви), и вторжения (intrusions). Для осуществления вторжения злоумышленником может быть использована внутренняя неисправность системы или внешнее воздействие. В данном случае для построения модели нас интересует сам факт наличия неисправностей в системе вследствие вторжения злоумышленника.

Принимая в качестве базового граф состояний системы, учитывающий возможное отсутствие информационного ресурса (рис. 1), можно построить граф включающий состояния системы, в которые она переходит в случае возникновения отказов по причине неисправностей программного/аппаратного обеспечения и атаки злоумышленника.

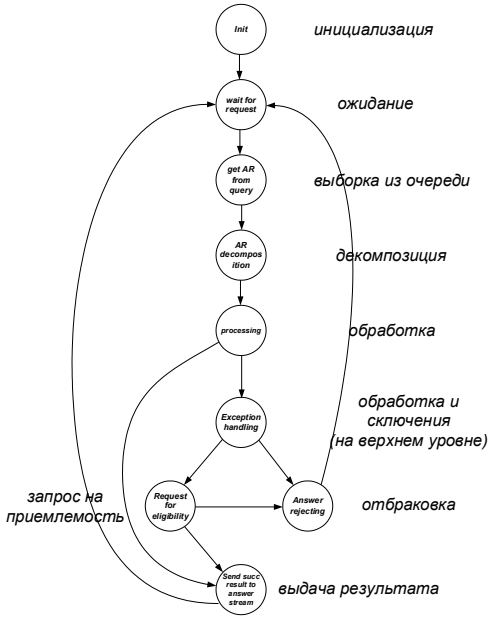

Рис. 1. Граф состояний системы с учетом отсутствия информационного ресурса

При разработке этого графа 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) прекра-

щает функционировать до полного восстановления и повторной инициализации.

Получение марковского графа. Модель (рис. 2) описывает все возможные состояния системы Web Services Upgrading Services [3]. Анализ модели показывает, что она не отвечает условиям марковости вследствие зависимости перехода системы в новое состояние от состояния, в котором она пребывала до перехода.

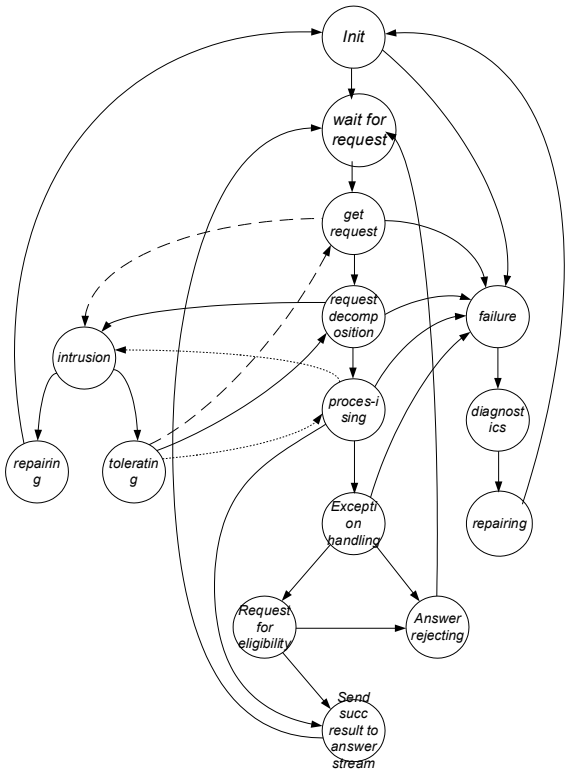

Рис. 2. Граф состояний системы с учетом воздействий

Если провести декомпозицию состояний, это приведет к незначительному увеличению количества состояний системы, однако исключит зависимость переходов. Модель системы после декомпозиции состояний показана на рис. 3.

В итоге получаем модель со следующими состояниями: *S*0 – инициализация системы; *S*1 – ожидание заявки; *S*2 – получение заявки из очереди; *S*3 – декомпозиция заявки; *S*4 – обслуживание заявки; *S*5 – исключение частично обслуженной заявки; *S*6 – исключение необслуженной заявки; *S*7 – запрос о принятии; *S*8 – удаление необслуженной заявки; *S*9 – отсылка результата обслуживания заявки; *S*10 – отказ системы по причине неисправности; *S*11 – диагностирование неисправности; *S*12 – восстановление системы; *S*13 – парирование вторжения в состоянии *S*2; *S*14 – парирование вторжения в состоянии *S*3; *S*15 – парирование вторжения в состоянии *S*4; *S*16 – отказ системы; *S*17 – восстановление системы после вторжения.

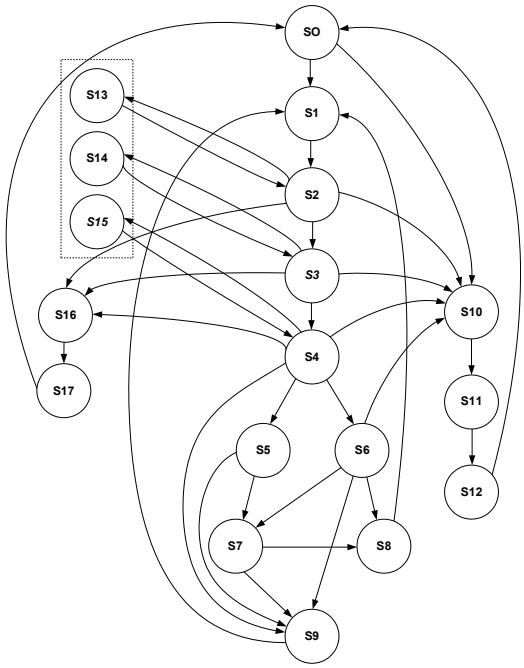

Рис. 3. Неразмеченный граф марковской модели системы

Обоснование выбора данных для определения параметров при разметке графа. Для расчета загрузки web-сервисов [4] принимается интенсивность запросов в интервале 1 – 100 запросов/сек (100 – 10.000 запросов / час).

Для каждого сервера, входящего в кластерную систему, интенсивность сбоя можно принять 1/час [5], а интенсивность отказа – 0,0001 1/час. Интенсивность отказа, вызванного ПО, можно считать на порядок меньше, чем интенсивность отказа аппаратуры, т.е. 10^{-5} 1/час. При последовательном соединении 100 серверов получаем интенсивность отказа 0,1 1/час. Согласно [3] интенсивность вторжений в начальный момент времени примерно составляет 0,01 – 0,1 % от потока входных заявок; в случае нахождения злоумышленником уязвимости, поток вторжений, атакующих уязвимость, может соста-

вить до 1 % от потока входных запросов.

Результаты моделирования. Принято, что в начальный момент времени система находится в состоянии *S*0 (инициализация). Как можно заметить, при заданных интенсивностях переходов система перейдет в стационарное состояние по истечению ~ 0,1 час. Самая высокая кривая с максимумом ~ 0,65 отображает вероятность нахождения системы в состоянии *S*3 (декомпозиция запроса). С точки зрения физики поведения системы это объясняется довольно большой длительностью декомпозиции входящего композитного запроса на подзапросы (рис. 4).

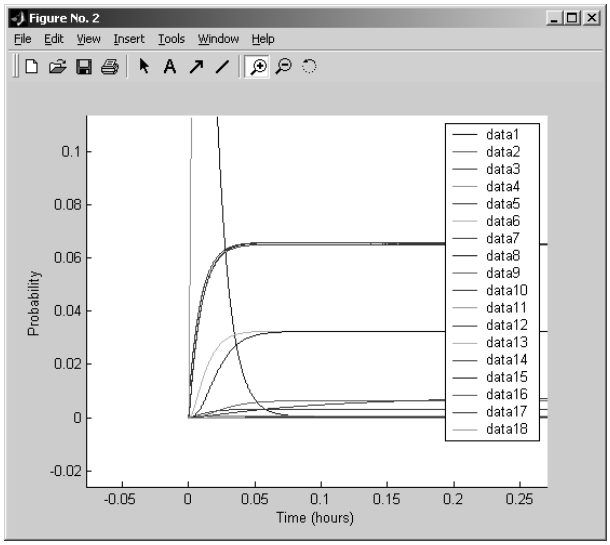

Рис. 4. График состояний системы на промежутке 0 – 0,25 ч

Заданные значения интенсивности парируемых и непарируемых вторжений обусловливают довольно высокое по сравнению с другими значение (~0,065 в стационарном состоянии) для состояний *S*13-*S*15 (парируемый отказ вследствие вторжения для состояний *S*2, *S*3, *S*4).

Система выдерживает увеличение интенсивности отказов в 10 раз без нарушения своего функционирования. Это объясняется достаточно высокой интенсивностью перехода в состояния диагностирования и восстановления после отказа. На приведенных графиках интенсивность перехода системы из состояния отказа в состояние диагностирования составляет 100 1/час, что очень хорошо характеризует систему с точки зрения оперативного вмешательства системного администратора в процесс восстановления системы после отказа. Если уменьшить эту интенсивность (*S*10->*S*11) в 10 раз – до значения 10 1/час, наблюдается «всплеск» кривой, которая описывает нахождение системы в состоянии декомпозиции запроса (рис. 5).

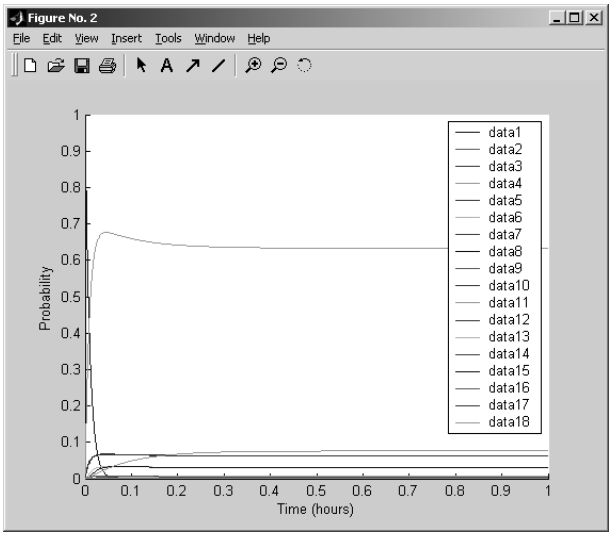

Рис. 5. Поведение системы в случае уменьшения в 10 раз интенсивности перехода в состояние диагностирования

Вероятность нахождения системы в состоянии отказа повышается, а вероятность обработки запроса снижается в тот момент, когда интенсивности диагностирования и последующего восстановления «хватает» на оперативное восстановление и перезапуск системы. Когда интенсивность диагностирования снижается еще в 10 раз, то система катастрофически теряет свою готовность. По прошествии двух часов с момента запуска системы, вероятность нахождения системы в состоянии отказа превышает вероятность нахождения системы в работоспособном состоянии. Таким образом, снижение интенсивности перехода в состояние диагностирования с 10 до 1 1/час является катастрофичным для системы.

Попробуем увеличить 3 интенсивности (L216, L316 и L416) в 10 раз, что будет соответствовать 1 1/час. График состояний системы в этом случае показан на рис. 6.

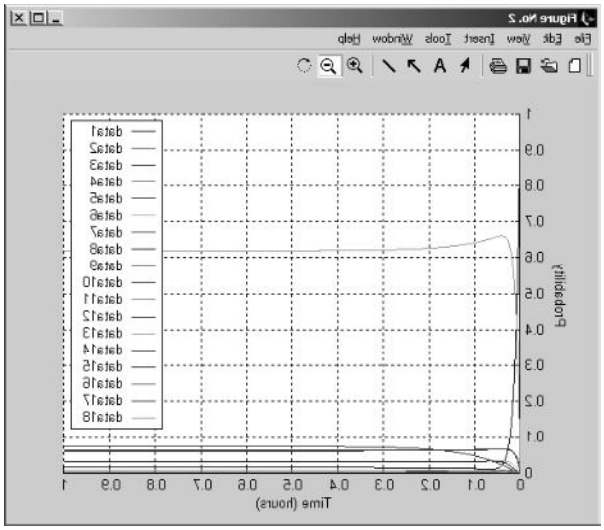

Рис. 6. Поведение системы при увеличенных в 10 раз интенсивностях перехода в состояние непарируемого вторжения

После непарированного вторжения система начинает проводить больше времени в состоянии *S*16 (отказ вследствие непарированного вторжения). Для возврата в первоначальное состояние системе требуется увеличить интенсивность восстановления после непарированного вторжения в 10 раз (до 100 1/час).

Следует отметить, что представленная модель не позволяет определить производительность системы, т.е. ее способность обработать некоторое количество запросов в единицу времени, а лишь оценивает вероятности нахождения системы в каждом из ее состояний в зависимости от интенсивностей переходов между состояний.

Заключение

В данной статье были рассмотрены аспекты функционирования системы управления обновлениями web-сервисов как системы массового обслуживания. Полученная модель функционирования системы позволяет рассчитать вероятность нахождения системы в каждом из ее состояний при заданных значениях интенсивности переходов. Чем меньше интенсивность восстановления после отказа (и восстановления после вторжений – как парированных, так и непарируемых), тем больше вероятность того, что система будет пребывать в этих состояниях, а значит, тем меньше готовность системы в целом. Проведенное исследование подтвердило факт, согласно которому в сложных системах гораздо проще уменьшить время восстановления (в нашем случае – увеличить интенсивность перехода из состояния отказа в работоспособное состояние), чем увеличить среднюю наработку на отказ.

Литература

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

2. 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 Development of Dependable Web Services out of Undependable Web Components, School of Computing Science, Univer. of Newcastle, 2004. – 36 p.

4. 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.

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

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

Поступила в редакцию 3.03.2006

Рецензент: д-р техн. наук, проф. Б.М. Конорев, Национальный аэрокосмический университет им. Н.Е. Жуковского "ХАИ", Харьков.