5. Рихтер Д., ван де Боспурт М. Программирование на C# для профессионалов. М.: Вильямс, 2014. 368 с.

6. Леоненков А. Самоучитель UML. 2-е издание. СПб.: «БХВ-Петербург», 2004. 576 с.

7. Бадд Т. Обьектно-ориентированное программирование в действии: Пер. с англ. СПб.: «Питер», 1997. 464 с.

УДК 004.855.5
doi:10.18720/SPBPU/2/id20-225

***Монахов Вадим Александрович***[1],
студент магистратуры;
***Магер Владимир Евстафьевич***[2],
канд. техн. наук, ст. науч. сотр., доцент

# МОДЕЛИРОВАНИЕ КАЧЕСТВА ИНФОРМАЦИОННЫХ СИСТЕМ

[1, 2] Санкт-Петербургский политехнический университет Петра Великого,
Санкт-Петербург, Россия,
[1] vadimmonakhov007@gmail.com, [2] mv@qmd.spbstu.ru

***Аннотация***. Зависимость от информационных систем постоянно растет, поэтому становится все более важным понимать качество системы, поскольку системы, обеспечивающие низкое качество обслуживания, порождают дорогостоящие последствия. Показано, что решение проблем с опозданием, например, после внедрения, является чрезмерно дорогостоящим, поскольку может включать в себя перестройку и повторную реализацию всей программной системы. Таким образом, важно анализировать качество программного обеспечения системы на ранней стадии, а именно, во время проектирования системы.

***Ключевые слова***: оценка качества, моделирование качества, надежность информационной системы.

***Vadim A. Monakhov***[1],
Master Student;
***Vladimir E. Mager***[2],
Candidate of Technical Sciences, Associate Professor

# MODELING OF INFORMATION SYSTEMS' QUALITY

[1, 2] Peter the Great St. Petersburg Polytechnic University,
St. Petersburg, Russia,
[1] vadimmonakhov007@gmail.com, [2] mv@qmd.spbstu.ru

***Abstract***. Dependence on information systems is constantly growing, so it is becoming important to understand the quality of the system, because systems that provide poor quality of service generate costly consequences. It is shown that solving problems

with delay, for example, after implementation, is excessively expensive, since it may involve rebuilding and re-implementing the entire software system. Thus, it is important to analyze the quality of the system software at an early stage, namely, during the design of the system.

***Keywords***: estimation of quality, modeling of quality, reliability of information system.

**Введение**

В априорном анализе качества программного обеспечения (ПО), в дополнение к анализу компонентов, которые разрабатываются с нуля, также необходимо проанализировать существующие компоненты, которые интегрируются в систему, поскольку разработчики ПО используют их для экономии затрат на разработку.

**1. Постановка задачи**

При решении данной задачи можно сфокусироваться на двух важных аспектах раннего анализа качества ПО: стоимость анализа и оценка параметров. Прежде чем их обсуждать, рассмотрим проблемное пространство в раннем анализе качества ПО, изображенное на рисунке 1.

По горизонтали указано количество информации, доступной для компонентов системы. Инженеры-программисты могут разрабатывать новые компоненты или интегрировать в систему существующие компоненты, возможно предоставленные сторонними производителями.

По вертикали отражена стоимость анализа. Хотя в случае доступности реализации методы, основанные на тестировании, могут быть применены, они являются дорогостоящими, поскольку включают выполнение большого числа запросов. Если исходный код или двоичные файлы доступны, могут применяться методы анализа программ и/или обратного инжиниринга. Несмотря на то, что эти методы имеют менее дорогостоящие, они не применимы, когда двоичные файлы доступны, но не могут быть подвержены анализу качества.

В предлагаемой информационной системе реализуется подход, основанный на сценариях. Идея имеет реализации, но все существующие решения имеют недостатки, поскольку не позволяют получить ответы на вопросы:

1) Как можно обосновать модель надежности, описывающей поведение сложной системы, состоящее из многосценарных последовательностей, если модель не масштабируется?
2) Как можно эффективно оценить надежность системы, если система состоит из десятков или сотен одновременно работающих компонентов и сценариев со схожим поведением?

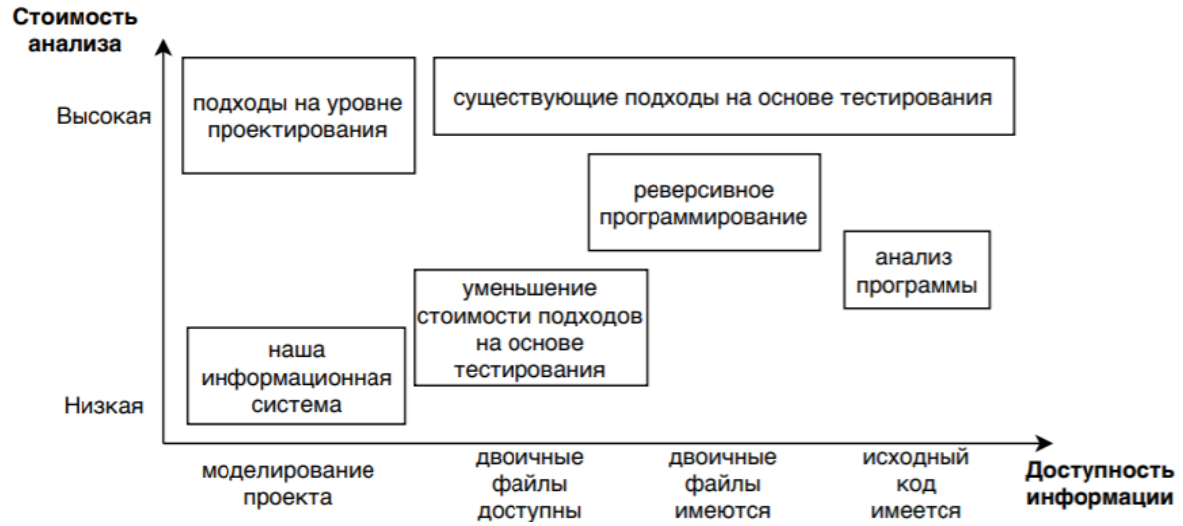



Рис. 1. Проблемное пространство в раннем анализе качества ПО

В предлагаемой системе определена возможность оперировать с последовательными комбинациями сценариев без необходимости анализировать полную модель системы, при этом не упрощая взаимосвязанность и независимость сценариев. Эта информационная система включает 3 процесса деятельности, предназначенных для базовых сценариев, сложных последовательных сценариев (далее – *ПОСЛ)* и сложных параллельных сценариев (далее – *ПАР)*. Каждый из этих процессов включает шаги для решения задач соответствующих моделей надежности и времени выполнения; последнее используется в *ПАР* сценариях. Вначале анализу подлежат базовые сценарии, и результаты этого анализа передаются на более сложные *ПОСЛ* и *ПАР* сценарии. Важными параметрами при вычислении моделей надежности, основанных на сценариях, являются: среднее время ожидания отправки запроса от компонента до компонента (когда запросов на принимающий компонент больше 1), результат передачи или приема запроса, который отмечается состоянием компонента *СОБЫТИЕ* или *СОСТОЯНИЕ ОТКАЗА* (обозначим их как *С* и *СО* соответственно).

Анализ модели надежности сценария будет основываться на условных вероятностях ненахождения компонента в *СО* при условии выполнения соответствующего сценария.

Прототипом предлагаемой информационной системы является системой сенсорной сети, реализованной с помощью фреймворка MIDAS, которая отслеживает температуру в комнате и включает или выключает кондиционер в соответствии с уникальными температурными пожеланиями пользователя (рис. 2). Воспользуемся ей для дальнейшей оценки.

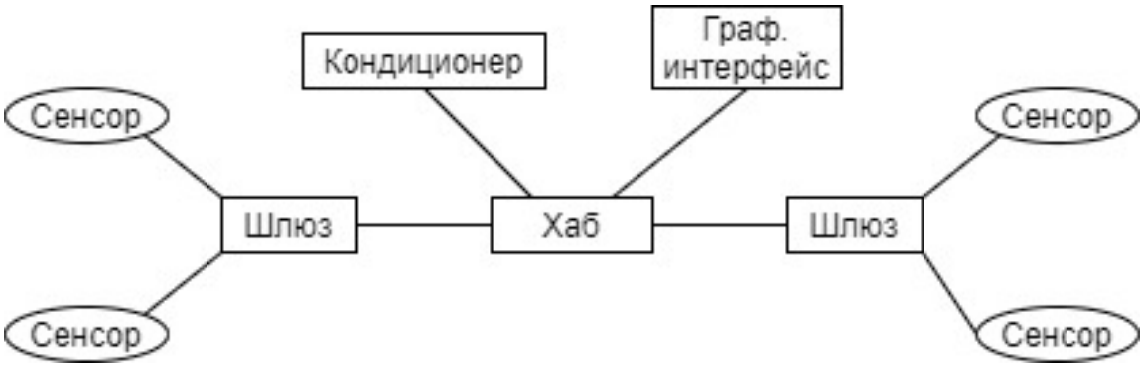



Рис. 2. Система MIDAS

Мы создаем модели надежности, основанные на сценариях (MHOC) по аналогии с существующими исследованиями [1] (*шаг 1.1*). Особенностью этого подхода является генерация состояния отказов для базовых сценариев на основе соответствия между моделями надежности компонента и сгенерированной MHOC. Таким образом удается повторно использовать информацию о дефектах архитектуры на уровне компонентов, что делает анализ надежности более полезным, а не заставлять инженера угадывать состояния отказа. Затем мы расширяем сгенерированную MHOC для моделирования конкуренции за ресурсы с помощью специальных состояний «постановки в очередь» (*шаг 1.2*). После того, как модель надежности MHOC построена и СМО для случая конкуренции решена, мы решаем MHOC для надежности сценария (*шаг 1.3*) и времени завершения (*шаг 1.4*), используя стандартные методы.

**2. Базовый сценарий**

Чтобы сгенерировать MHOC для сценария, вначале требуется сгенерировать подмодель для каждого компонента в каждом сценарии, а затем применить параллельную композицию.

Для примера MIDAS компоненты модели для сценария *СенсорШлюз* изображены на рисунке 3. В подмодели также включены состояния отказа (-1) и возможные переходы в состояние отказа и из него.

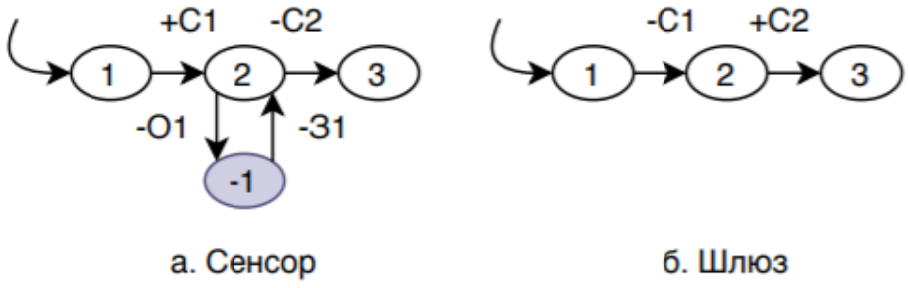



Рис. 3. Подмодели компонента *СенсорШлюз*

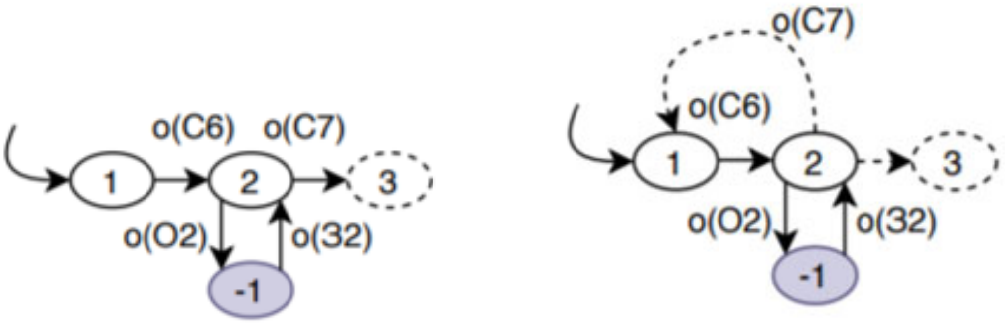
Дополнительной информацией, включаемой в МНОС, являются результаты моделирования конфликтов, а именно: состояние очереди $q$, определяемое как

$$Q(j,q) = q(E) \text{ и } Q(q,k) = q(ready),$$

где $E$ – событие, инициирующее переход из состояния $j$ в состояние $k$; *ready* – событие, когда отвечающий готов обработать запрос вызывающего.

Последнее определяется как $q(ready) = \frac{1}{T_{wait}}$, где $T_{wait}$ – среднее время ожидания вызывающего. Для вычисления $T_{wait}$ решается сеть массового обслуживания, которая описывает поведение запросов в очереди.

Для поиска надежности базового сценария полагаем, что система работает в течение длительного времени, и что после выполнения анализируемого сценария $Scen_i$ она в итоге снова его выполнит. Выполнение сценария заканчивается, когда он собирается перейти в *Конечное* состояние, которое представляет начало другого сценария. Поэтому мы убираем *Конечное* состояние и перераспределяем скорость перехода в начальное состояние. К примеру, в МНОС для *ГИПЗапрос* (*запрос к графическому интерфейсу*) на рисунке 4.а мы заменяем переход из состояния 2 в состояние 3 переходом из состояния 2 в состояние 1, при скорости перехода $q(07)$, как показано на рисунке 4.б.




а. *ГИПЗапрос* б. Переход из состояния 2 в состояние 1

Рис. 4. Оценка перераспределения для *ГИПЗапрос*

Теперь эту модель можно решить для установившегося вектора вероятности состояний $\vec{\pi}_i$, тогда надежность сценария вычисляется следующим как

$$r_i = 1 - \sum_{f \in F_i} \pi_i(f), \quad (1)$$

где $F_i$ – набор состояний отказа в МНОС для $Scen_i$. Матрица оценки модели *ГИПЗапрос* после перераспределения:

$$\begin{matrix} 1 \\ 2 \\ -1 \end{matrix}\begin{bmatrix} -0.005 & 0.005 & 0 \\ 0.1 & -0.11 & 0.01 \\ 0 & 0.4 & -0.4 \end{bmatrix} \quad (2)$$

Решение этой матрицы с использованием стандартного метода дает: $\vec{\pi}_i = \begin{bmatrix} 0.9512 & 0.0476 & 0.0012 \end{bmatrix}$, следовательно, надежность сценария *ГИПЗапрос* составляет $r_i = 1 - 0.0012 = 0.9988$.

Для обеспечения условия параллелизма в *ПАР* сценарии необходимо определить время выполнения базового сценария. Пусть $T_i(S)$ – это время выполнения, когда система находится в состоянии $i$ из МНОС для $Scen_i$, т. е. $T_i(1)$. Вычислим $\vec{T}_i$, применив анализ переходных процессов, которые соответствуют решению уравнения (3):

$$Q_i' \cdot \vec{T}_i = -e, \quad (3)$$

где $Q_i'$ – матрица после удаления строк и столбцов, отвечающих за конечное состояние в матрице $Q_i$, $e$ –вектор-столбец содержащий -1 и имеющий соответствующие размерности. Согласно сценарию *ГИПЗапрос*, матрица оценок $Q_i'$ определяется уравнением (2).

Применив уравнение (3) к $Q_i'$, для *ГИПЗапроса* получим: $\vec{T}_i = \begin{bmatrix} 201.03 & 1.025 & 3.525 \end{bmatrix}$, из чего следует $t_i$ = 201.03. Значения $t_i$ для MIDAS приведены в таблице 1.

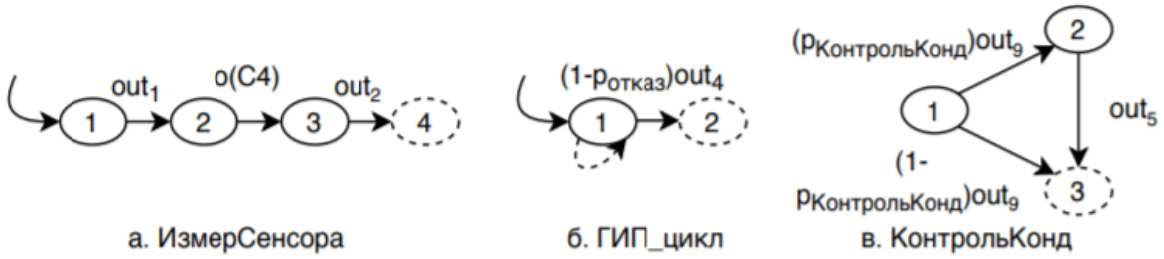
*Таблица* 1

**Значения $r_i$ и $t_i$ для MIDAS сценария**

| Сценарий | $r_i$ | $t_i$ | Сценарий | $r_i$ | $t_i$ |
|---|---|---|---|---|---|
| СенсорШлюз | 0,9900 | 6,2625 | ШлюзХаб | 1 | 15 |
| ШлюзПодт | 1 | 2 | Сенсоры_ПАР | 0,9876 | 9,3937 |
| ШлюзПодт_ПАР | 1 | 3 | ИзмерСенсора | 0,9876 | 27,394 |
| ГИПЗапрос | 0,9999 | 201,03 | ИзмнТемпКонд | 1 | 0,5 |
| ГИП_цикл | 0,9999 | 205,13 | КонтрольКонд | 0,9999 | 205,28 |
| Система | 0,9940 | 287,46 | --- | --- | --- |

### 3. Последовательные сценарии

При их анализе применим стохастическое дополнение, скомбинировав подсценарии МНОС, входящие в *ПОСЛ* сценарий. Для этого вначале генерируем состояния модели (подсценарии), а затем вычисляем скорости перехода относительно примененного стохастического дополнения. Если *ПОСЛ* сценарий $Scen_i$ имеет подсценарий $Scen_k$, выполняемый после другого подсценария $Scen_i$, добавляеся переход из состояния $j$ в состояние $k$ в МНОС для $Scen_i$. МНОС для сценария *ПОСЛ* в MIDAS изображены на рисунке 5.




а. ИзмерСенсора б. ГИП_цикл в. КонтрольКонд

Рис. 5. МНОС для *ПОСЛ* сценария

Скорости для каждого перехода, определенного выше, вычисляются как:

$$Q_i(j,k) = [p_i(j,k)] \cdot out_j, \tag{4}$$

где $p_i(j,k)$ – это вероятность выполнения $Scen_k$ после выполнения $Scen_j$, $out_j$ определяется как:

$$out_j = \sum_{s \in S_i} [p_j(s)] \cdot Q_i(s, End), \tag{5}$$

где $S_j$ – набор состояний в сценарии $Scen_j$, $p_j(s)$ – вероятность устойчивого состояния нахождения в состоянии $s$ в $Scen_j$, $Q_j(s,End)$ –оценка перехода из состояния $s$ в конечное состояние в $Scen_j$.

Например, для *ГИП_цикл*, показанного на рис. 5.б, $Scen_i$ = *ГИП_цикл* и $Scen_j$ = *ГИПЗапрос*, и оценки перехода из состояния 1 в состояние 2 могут быть вычислены по формуле:

$$Q_i(1,2) = P_i(1,2) \cdot \left\{ \sum_{s \in S_j} [p_i(s) \cdot Q_j(s, End)] \right\} \tag{6}$$

На основании (6) можно рассчитать оценку перехода из состояния 1 в состояние 2 для $Scen_i$:

$$Q_i(1,2) = (1 - P_{отказ}) \cdot [p_j(2)] \cdot [q(E7)] = 0.98 \cdot 0.005 \cdot 1 = 0.0049.$$

Когда мы перемещаемся вверх по уровню иерархии к *КонтрольКонд*, оценка перехода из состояния 1 в состояние 2 в МНОС для *КонтрольКонд* на основании (6) будет следующей:

$$Q_i(1,2) = (P_{КонтрольКонд}) \cdot (1) \cdot [Q_j(1, End)] = 0.3 \cdot 1 \cdot 0.0049 = 0.0015.$$

***Надежность*** последовательного сценария вычисляется аналогично базовому. После вычисления $\vec{p}_i$ и $r_j$ для всех подсценариев в $Scen_j$, надежность этого сценария рассчитывается на основе уравнения:

$$r_i = 1 - \sum_j p_i(j) \cdot r_j. \tag{7}$$

В рассматриваемом примере для определения $\vec{p}_i$ воспользуемся МНОС для *КонтрольКонд*, и после перераспределения скорости перехода в конечное состояние получим следующую матрицу скоростей:

$$\begin{bmatrix} -0.0015 & 0.0015 \\ 2 & 2 \end{bmatrix}.$$

Решая эту модель, получаем: $\vec{p}_i$ = [0.9993 0.0007]. Надежность *ИзмнТемпКонд* равна 1, поэтому в данном сценарии дефекты отсутствуют. Следовательно, надежность сценария *КонтрольКонд* состовляет: $r_i$ = 0.993·0.9999 + 0.0007·1 = 0.9999. Надежность остальных сценариев, показанных в табл. 1, вычисляются аналогично.

***Время выполнения*** последовательного сценария состоит из объединения времени выполнения подсценариев. Для примера рассмотрим сценарий *ГИП_цикл*. Матрица скоростей для вычисления времени выполнения с обновлением скоростей вычисляется следующим образом. Пусть $Scen_i$ = *ГИП_цикл*, а $Scen_j$ = *ГИПЗапрос*, тогда, используя матрицу скоростей и вероятностный вектор установившегося состояния, указанный выше, можно обновить скорость перехода из состояния 1 в состояние 2 как:

$$Q_i(1,2) = p_i(1,2) \cdot \frac{1}{t_j} = (1 - 0.02) \cdot \frac{1}{201.03} = 0.0049.$$

Применив формулу (3), и рассчитываем время выполнения *ГИП_цикл*: 205,12 ед. времени.

Следующим уровнем вверх по иерархии является сценарий *КонтрольКонд*. Пусть $Scen_i$ = *КонтрольКонд*, и $Scen_j$ = *ГИП_цикл*, тогда скорость перехода из состояния 1 в состояние 2 в *КонтрольКонд* равна

$$Q_i(1,2) = p_{1,2} \cdot \frac{1}{t_j} = p_{КонтрольКонд} \cdot \frac{1}{t_j} = 0.3 \cdot \frac{1}{205.12} = 0.0015.$$

Аналогично, скорость перехода из состояния 1 в состояние 3 составляет:

$$Q_i(1,3) = p_{1,3} \cdot \frac{1}{t_j} = (1 - p_{КонтрольКонд}) \cdot \frac{1}{t_j} = 0.7 \cdot \frac{1}{205.12} = 0.0034.$$

Скорость перехода из состояния 2 в состояние 3 – это скорость завершения *ИзменТемпКонд*, которая равна $Q_i(2,3) = \frac{1}{0.5} = 2.$ С учетом этих параметров матрица скоростей модели $Q'$ (рис. 5) выглядит следующим образом:

$$Q' = \begin{matrix} 1 \\ 2 \end{matrix} \begin{bmatrix} -0.0047 & 0.0015 \\ 0 & -2 \end{bmatrix} \qquad (8)$$

Таким образом, скорость завершения *КонтрольКонд* после решения $Q'$ с использованием уравнения 3 равна: $t_i = 205.28$ ед. времени.

***Шаги моделирования ПАР сценария**:*

1) определение возможных комбинаций сценариев,
2) создание модели,
3) усечение модели,
4) вычисление вероятностей комбинаций,
5) вычисление надежностей комбинаций,
6) вычисление надежности сценария,
7) вычисление времени выполнения.

Описание вычисления параллельных сценариев заслуживает отдельного внимания, но это описание слишком громоздко, поэтому будет опубликовано в следующей статье.

**Заключение**

Описаны основы предлагаемой информационной системы, базирующейся на понятии сценариев, состояний компонентов и их переходов между состояниями в системе, качество которой оценивается.

Для каждого типа сценариев указаны шаги и приведено описание этих шагов, необходимых для моделирования сценариев и вычисления их надежности и времени выполнения.

Дальнейшее применение и последующая оценка предлагаемой системы демонстрируют положительные результаты, показывают преимущества в сравнении с аналогичными подходами. Так, например, предлагаемая система обладает хорошей масштабируемостью, и при увеличении числа компонентов или базовых сценариев не проигрывает в точности. Другим преимуществом является тот факт, что даже при большой масштабируемости вычислительные ресурсы и стоимость вычислений остаются практически на прежнем же уровне, за счет примененного усечения моделей без потери точности в прогнозировании и оценки надежности.

**Список литературы**

1. Yacoub S., Cukic B., Ammar H.H. A scenario-based reliability analysis approach for component-based software // Proc. of the IEEE Transactions on Reliability. December 2004. Vol. 53(4). Publisher: IEEE. P. 465–480. DOI: 10.1109/TR.2004.838034.

2. Stewart W. Probability, Markov Chains, Queues, and Simulation: The Mathematical Basis of Performance Modeling. Princeton, New Jersey, USA: Princeton University Press, 2009. 776 p.