На правах рукописи

Дубаков Сергей Анатольевич

Информационная технология анализа производительности в процессе разработки программного обеспечения

Специальность: 05.13.11 – Математическое и программное обеспечение вычислительных машин, комплексов и компьютерных сетей

АВТОРЕФЕРАТ

диссертации на соискание ученой степени кандидата технических наук

Томск – 2005

Работа выполнена в Томском политехническом университете

Научный руководитель:	доктор технических наук, профессор Силич В.А.
Оффициальные оппоненты:	Марков Н.Г. – д.т.н., профессор кафедры ВТ ТПУ, г. Томск Максимов А.В. – канд. физ.-мат.наук , доцент, зав. каф. информатики АГУ, г. Барнаул

Ведущая организация: Томский Университет Систем Управления и Радиоэлектроники, г. Томск

Защита состоится « 21 » декабря 2005 г. В 15 час. 00 мин. на заседании диссертационного совета Д 212.269.06 по адресу: 634034, г.Томск, ул. Советская, 84, институт «Кибернетический центр» ТПУ.

С диссертацией можно ознакомиться в библиотеке Томского политехнического университета по адресу: 645045, г.Томск, ул. Белинского, 53.

Автореферат разослан « __ » ноября 2005 г.

Ученый секретарь
диссертационного совета
к.т.н., доцент М.А. Сонькин

Общая характеристика работы

Актуальность исследования. Одной из ключевых характеристик современного программного обеспечения (ПО) является производительность. Низкая производительность может обуславливать неэффективность работы пользователей и, как следствие, негативную реакцию на программный продукт, потерю прибыли и клиентуры, доли рынка; перерасход средств вследствие затрат на модификации и перепроектирование. Более того, переработка кода с целью увеличения производительности с большой вероятностью может разрушить первоначальную структуру системы, сводя на нет преимущества от использования объектно-ориентированного подхода. Наконец, маловероятно, что переработанный код сможет довести производительность системы до уровня, на котором она могла бы быть в случае изначального проектирования с учетом вопроса производительности. В худшем случае, будет невозможно достичь поставленных целей простой модификацией исходного кода системы, что обусловит необходимость полного перепроектирования или остановки проекта.

Очевидно, что большинство серьезных проблем с производительностью ПО вызвано недооценкой этого вопроса на ранних стадиях процесса разработки, в частности на этапе проектирования. Низкая производительность чаще всего обусловлена недостатками проектирования, а не реализации. Однако, в объектно-ориентированном сообществе распространена тенденция откладывать рассмотрение вопроса производительности до момента, когда система реализована.

Подобные взгляды базируются на представлении о том, что производительность ПО трудно спрогнозировать, и модели, необходимые для прогнозирования производительности разрабатываемой системы слишком сложны и требуют значительных затрат. Помимо этого, сложившаяся практика такова, что область проектирования ПО с одной стороны и анализа производительности с другой - являются достаточно удаленными, практически изолированными, друг от друга.

Несмотря на указанные проблемы, многие эксперты считают, что спроектировать программную систему с учетом вопросов производительностью практически возможно. Этой задаче посвящены работы таких авторов, как Смит К., Уильямс Л., Петриу Д., Вудсайд М., Хиллстон Дж., Ролиа Дж., Фрэнкс Р. Регулярно проводятся международные конференции, посвященные вопросам анализа производительности ПО, в частности International Workshop on Software and Performance (WOSP).

Слабым метом существующих решений по анализу производительности программного обеспечения является необходимость создания и поддержки отдельной модели производительности в дополнение к уже существующим моделям (таким как модель проектирования, модель данных и т.д.). Кроме того, большинство подобных подходов рассчитано на непосредственное использование специализированных математических моделей, что повышает требование к

подготовке участников команды разработки и, тем самым, снижает эффективность внедрения.

Таким образом, несмотря на активные разработки в данной области, проблема создания эффективного подхода к анализу производительности в процессе разработки ПО до конца не решена. Для того, чтобы было возможно учитывать вопросы производительности в течении всего процесса разработки, начиная с ранних стадий, должна быть создана технология, позволяющая, не нарушая сложившийся подход к проектированию ПО, оценивать те или иные архитектурные решения с точки зрения производительности. При этом должны быть минимизированы издержки использования подобной технологии, для чего она должна опираться в основном на уже существующие артефакты проектирования и не требовать математической подготовки от участников коллектива разработчиков. Под артефактами проектирования понимаются данные моделирования, управления проектами, системной конфигурации, т. е. компоненты информационных систем, которые описывают информационные системы.

Цель работы: разработка информационной технологии анализа производительности в процессе разработки ПО, обеспечивающей повышение качества и снижение затрат на его создание. Для достижения поставленной цели были поставлены и решены следующие **задачи**:

1. Формулирование требований к технологии анализа производительности в процессе разработки ПО.
2. Анализ существующих технологий анализа производительности в процессе разработки ПО, обоснование необходимости разработки информационной технологии, удовлетворяющей поставленным требованиям.
3. Анализ существующих средств моделирования производительности ПО.
4. Разработка информационной технологии анализа производительности в процессе разработки ПО.
5. Разработка программного комплекса, реализующего предложенный подход.

Методы исследования. В ходе диссертационного исследования были использованы метод экспертных оценок, объектно-ориентированное проектирование и программирование, математическое моделирование, аппарат многоуровневых сетей массового обслуживания, методы математической статистики.

Научная новизна. Получены следующие основные результаты, обладающие научной новизной:

1. Разработана новая информационная технология, позволяющей внедрить анализ производительности в жизненный цикл разработки ПО с ранних стадий без необходимости привлечения специализированных моделей и средств.

2. Предложен алгоритм автоматической генерации модели производительности на основе диаграмм UML.
3. Модифицирован алгоритм анализа многоуровневых сетей массового обслуживания для эффективной поддержки моделей, полученных на основе диаграмм UML.

Практическая значимость. Разработанная информационная технология позволяет внедрить превентивный подход к вопросам производительности в процесс разработки ПО без необходимости привлечения специализированных моделей и средств, обеспечивая повышение качества и снижение затрат на создание ПО.

Реализовано программное решение, автоматизирующее формирование и анализ модели производительности на основе уже существующих артефактов проектирования. Применение продукта не требует от участников команды разработки каких-либо дополнительных знаний в области моделирования производительности.

Реализация результатов и их внедрение. Разработанная информационная технология и поддерживающее ее программное решение внедрены в процессе разработки ПО в ООО «ТВ-Система» (г.Томск), ООО «Интрайс» (г.Томск), Томском Политехническом Университете (г.Томск).

Основные положения, выносимые на защиту:

1. Информационная технология анализа производительности в процессе разработки ПО.
2. Алгоритм автоматической генерации модели производительности средством многоуровневых сетей массового обслуживания на основе модели UML.
3. Алгоритм анализа многоуровневых сетей массового обслуживания, модифицированный для эффективной поддержки моделей, полученных на основе диаграмм UML.
4. Программное решение поддержки предложенной информационной технологии, позволяющее выполнять формирование и анализ модели производительности на основе диаграмм UML.

Апробация работы. Основные положения диссертации докладывались и обсуждались на научных семинарах кафедры оптимизации систем управления АВТФ ТПУ; конференции Ассоциации научных и учебных организаций-пользова телей сетей передачи данных «RELARN» (Нижний Новгород, 1997); всероссийской научно-методической конференции «Телематика» (1998); международной научно-практической конференции "Актуальные проблемы информатики и информационных технологий" (Тамбов, 2004); международном симпозиуме "KORUS" (Новосибирск, 1999; Томск, 2004); международной научной конференции "Инновации в науке и образовании" (Калининград, 2004); всероссийской научно-практической конференции "Системы и средства автоматизации" (Томск, 2004; Томск, 2005); всероссийской научно-технической конференции «Проблемы информатики в образовании, управлении, экономике и технике» (Пенза, 2005).

По результатам выполненных исследований опубликовано 11 работ.

Личный вклад. Постановка задач исследования выполнена автором совместно с В.А. Силичем. Все результаты, составляющие основное содержание диссертации, получены автором самостоятельно.

Структура и объем работы. Диссертация состоит из введения, 3 глав и заключения. Объем диссертации составляет 132 страниц текста, 45 рисунков, список литературы из 117 наименований.

Содержание работы

Во введении обосновывается актуальность темы диссертации, формулируется цель и задачи работы, ее научная новизна и практическая значимость. Приводится краткое содержание работы по главам.

В первой главе рассматривается проблема производительности как одной из ключевых характеристик ПО. Определены метрики и методы оценки производительности, включающие измерение и моделирование. Классифицированы издержки, вызываемые проблемами с производительностью.

Показана необходимость интеграции анализа производительности в процесс разработки ПО для гарантированного выполнения требований по производительности конечным продуктом, сформулированы требования к подобному подходу.

Проведен анализ существующих технологий анализа производительности в процессе разработки ПО, таких как Software Performance Engineering (SPE), Performance Design & Engineering (PDE), Structure & Performance Specification (SP), Hierarchical Evaluation Tool (HIT). Обоснована необходимость создания новой информационной технологии анализа производительности в процессе разработки ПО.

В рамках анализа требований к информационной технологии анализа производительности в процессе разработки ПО рассмотрены существующие средства моделирования производительности ПО, такие как сети Петри, сети массового обслуживания, алгебры процессов и имитационное моделирование.

Для оценки и выбора наиболее эффективного средства построения и анализа модели производительности была сформирована система критериев. Критерии были классифицированы на три группы: функциональные возможности, затраты на моделирование и анализ, наличие алгоритмов и средств анализа. В соответствии с важностью для применимости в информационной технологии каждому критерию назначен вес по шкале от 0 до 5.

Результаты оценки показали, что оптимальным средством построения модели производительности в можно считать многоуровневые сети массового обслуживания (Layered Queueing Networks – LQN). Этот формализм обладает достаточной выразительностью для представления сложных, в том числе многоуровневых и распределенных программных приложений. Он также выгодно отличается от других средств аналитического моделирования наличием эффективных методов анализа, таких как стохастические сети с рандеву

(Stochastic Rendezvous Networks) и метод слоев (Method of Layers). Эти методы не базируются на марковских цепях и, соответственно, не подвержены эффекту взрыва пространства состояний.

Обоснован выбор нотации UML в качестве средства описания исходных данных для информационной технологии анализа производительности в процессе разработки ПО. Рассмотрены существующие подходы к использованию UML для моделирования производительности.

Во второй главе рассматривается авторский подход к моделированию производительности на основе диаграмм UML с помощью многоуровневых сетей массового обслуживания.

В связи с тем, что стандартная модель UML не включает полный набор информации, необходимой для построения модели производительности, было предложено дополнение к нотации UML, включающее в себя следующие элементы:

- Для актеров (Actor) на диаграмме вариантов использования (Use Case Diagram):
  - Помеченное значение “multiplicity”, определяющее количество взаимодействующих с системой пользователей данного типа;
  - Помеченное значение “delay”, определяющее задержку между обращениями пользователя к системе;
- Для ассоциации между актером и вариантом использования (Use Case) на диаграмме вариантов использования:
  - Помеченное значение “multiplicity”, определяющее количество вызовов функции за один сеанс взаимодействия пользователя с системой;
- Для операции класса (Class) или интерфейса (Interface) на диаграмме классов (Class Diagram):
  - Помеченное значение “executionTime”, определяющее среднее прогнозируемое время выполнения метода;
- Для узлов (Node) на диаграмме развертывания (Deployment Diagram):
  - Помеченное значение “capacity”, определяющее вычислительную мощность аппаратного ресурса;
- Для ассоциаций между узлами на диаграмме развертывания:
  - Помеченное значение “communicationDelay”, определяющее пропускную способность коммуникационного канала между аппаратными ресурсами.

Предложен алгоритм формирования модели производительности с помощью многоуровневых сетей массового обслуживания на основе модели UML, блок-схема которого в виде диаграммы деятельности UML показана на рис. 1.

Шаг 1. Для каждого актера из диаграммы вариантов использования, взаимодействующего с каким-либо из выбранных для рассмотрения вариантов использования, создаются начальные задачи (reference tasks). Начальная задача в нотации LQN характерна тем, что не предоставляет услуг другим задачам, а выступает только в качестве клиента.

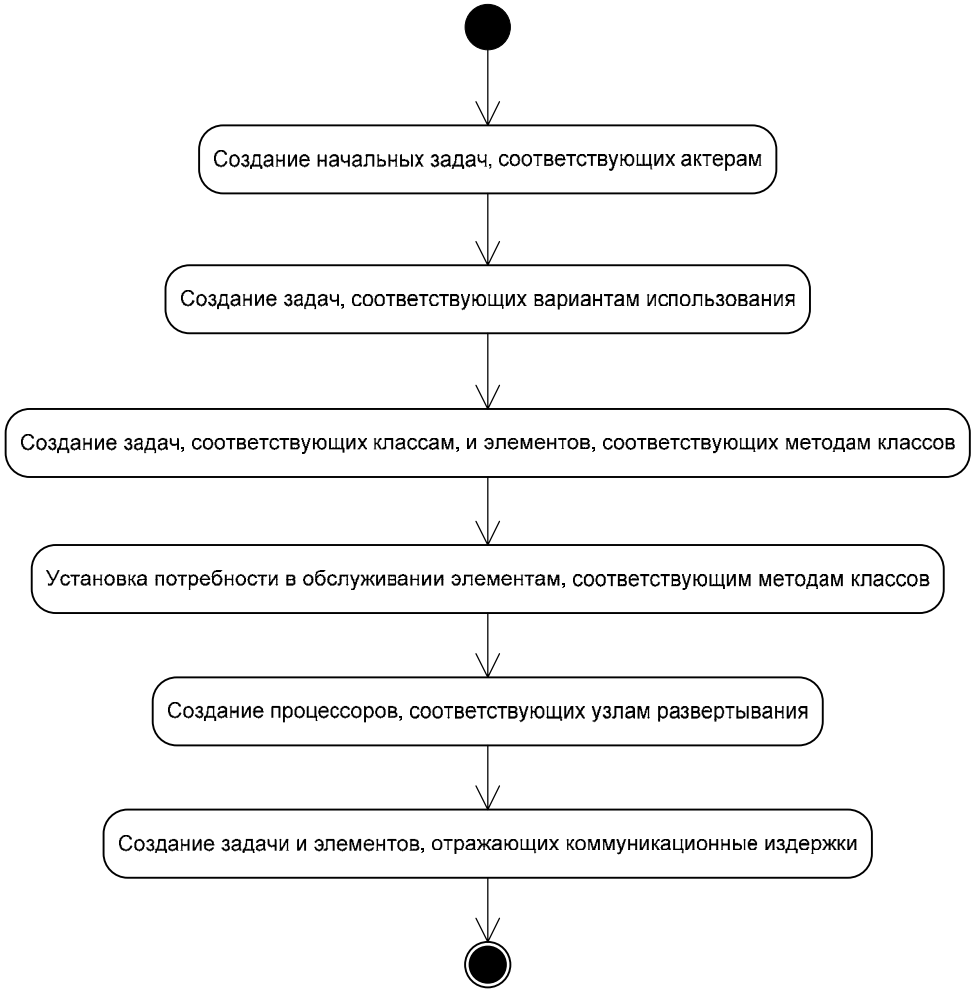

Рис.1. Блок-схема алгоритма построения модели производительности в виде диаграммы деятельности UML.

Шаг 2. Модель производительности дополняется задачами типа "активный сервер" (active server), соответствующими выбранным вариантам использования. Активные серверы в нотации LQN могут как выполнять запросы, поступающие от других задач, так и сами осуществлять запросы.

При добавлении вариантов использования в модель производительности, нужно учитывать возможность существования следующих взаимосвязей между ними:

- обобщение;
- включение;
- расширение.

Шаг 3. Модель производительности дополняется задачами, соответствующими классам, с помощью которых реализована анализируемая функциональность.

Для этого необходимо рассмотреть диаграммы последовательности, связанные с каждым вариантом использования, включенным в модель производительности.

На диаграмме последовательности по оси X расположены объекты, участвующие в реализации рассматриваемой функциональности. Для того, чтобы

учитывать их физическую реализацию каждый из них должен представлять собой экземпляр класса, определенного в модели UML. Таким образом, одновременно с рассмотрением диаграммы последовательности необходимо анализировать соответствующие диаграммы классов.

Шаг 4. После добавления задач, соответствующих классам, реализующим функциональность системы и элементов, соответствующих их методам, необходимо определить потребность в обслуживании (service demand) каждого элемента. Потребность в обслуживании определяется средним временем выполнения соответствующего метода соответствующего класса без учета времени выполнения вызываемых методов. Это значение можно получить из помеченного значения "executionTime" этого метода на диаграмме классов.

Шаг 5. Модель производительности дополняется процессорами, представляющими аппаратные ресурсы окружения системы.

Исходными данными для выполнения этого шага являются диаграммы компонентов и развертывания.

Каждый класс, представленный на модели производительности, должен входить в один из компонентов на диаграмме компонентов. Компоненты обеспечивают высокоуровневую группировку функциональности системы.

В свою очередь, диаграмма развертывания показывает на каком узле размещен тот или иной компонент. Узел однозначно отображается в процессор на модели производительности. Добавленный процессор связывается с задачей, соответствующей рассматриваемому классу.

Шаг 6. Кроме собственно ресурсов аппаратного окружения системы диаграмма развертывания содержит информацию о затратах на взаимодействие между компонентами, в случае если они размещены на различных узлах системы. Связи между узлами могут содержать помеченное значение "communicationDelay", отражающее среднее время, затрачиваемое на один удаленный вызов между этими узлами.

Для того, чтобы учесть подобные коммуникационные издержки на модели производительности, добавляется задача "Network" с отдельным процессором. Для каждой связи между двумя узлами диаграммы развертывания, создается элемент этой задачи с потребностью в обслуживании, равной помеченному значению "communicationDelay". После этого для каждой пары взаимодействующих элементов, задачи которых работают на различных процессорах, в вызывающий элемент добавляется обращение к задаче, соответствующей затратам на удаленный вызов между этими двумя процессорами. В случае если элемент делает несколько вызовов к элементам задачи или задач, работающих на одном процессоре, связь с элементом задачи "Network" параметризуется количеством таким вызовов.

В результате выполнения алгоритма формируется модель производительности в нотации многоуровневых сетей массового обслуживания, подобная изображенной на рис. 2.

Далее в главе рассмотрен анализ модели производительности в нотации LQN для расчета требуемых метрик. Для этого можно использоваться одним из

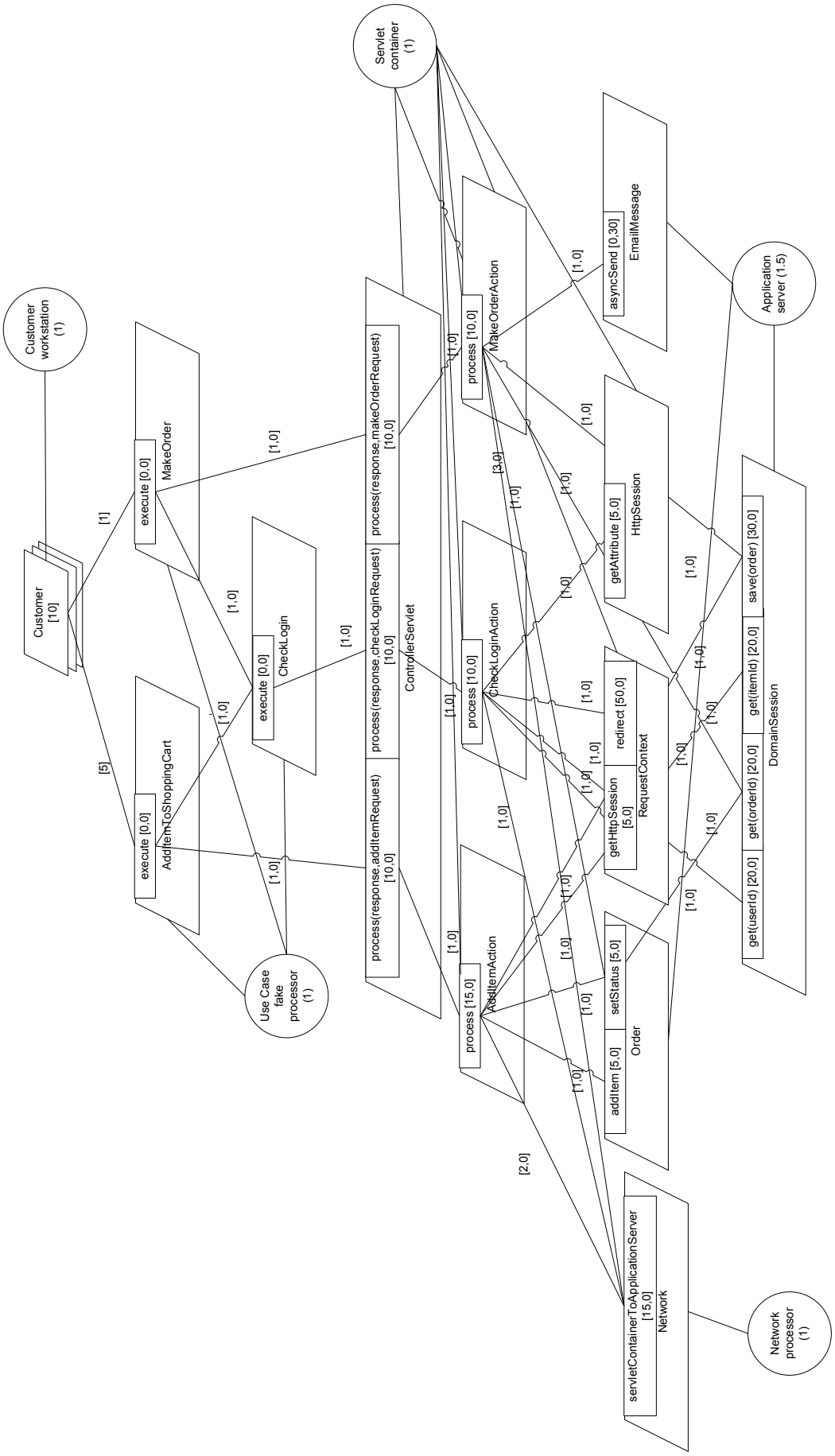

Рис. 2. Модель производительности в нотации LQN.

существующих методов анализа многоуровненвых сетей массового обслуживания. Метод Эгравала и Базена "Метод сгруппированных серверов" (Aggregate server method) может использоваться для анализа сетей только с одним уровнем серверов. Метод Ролиа, названный "Методом слоев" (Method of Layers) имеет меньше ограничений, так как поддерживает неограниченное количество уровней серверов, однако требует жесткого разделения на слои при котором задачи могут взаимодействовать только с задачами, находящимися на уровень ниже. И, наконец, наиболее функциональный метод, лишенный всех вышеупомянутых ограничений, был разработан Вудсайдом - это метод стохастических сетей с рандеву (Stochastic Rendezvous Networks - SRN). В данной работе использована одна из модификаций метода SRN, предложенная Роем Грегори Фрэнксом.

Суть метода, предложенного Фрэнксом, состоит в том, что многоуровневая сеть массового обслуживания разбивается на несколько обычных сетей, для анализа которых могут использоваться стандартные алгоритмы, такие как метод анализа среднего значения (Mean Value Analysis - MVA). Анализ сетей проводится в несколько итераций до достижения сходимости одного из результирующих значений.

В рамках рассматриваемой информационной технологии данный метод анализа дополнен процедурой редукции модели, специфичной для модели производительности, сформированной на основе диаграмм UML.

Блок-схема алгоритма в виде диаграммы деятельности UML представлена на рис. 3.

Редукция модели предназначена для максимального упрощения модели, полученной путем автоматической генерации на основе диаграмм UML. На этом шаге из модели удаляются избыточные задачи и процессоры.

Трансформация модели – это шаг, необходимый для подготовки модели многоуровневой сети массового обслуживания к анализу с помощью метода SRN. На нем все процессоры заменяются адекватной им задачей.

Разбиение на подмодели выполняется для разделения многоуровневой сети на несколько обычных сетей массового обслуживаний, которые можно затем анализировать методом MVA. Существует несколько возможных алгоритмов разбиения:

- строгое разбиение (Strict layering);
- свободное разбиение (Loose layering);
- пакетное разбиение (Batched layering);
- сжимающее разбиение (Squashed layering).

После этого выполняется основная часть алгоритма – анализ подмоделей.

Часть параметров сетей массового обслуживания является постоянными и устанавливаются при начальной инициализации. Сюда входят количество обращений посетителей сети к приборам обслуживания, а также численность посетителей каждого типа. Кроме того, неизменяемыми параметрами являются задержка потока заявок, соответствующего начальным задачам, а также время обслуживания приборов, соответствующих чистым серверам. Остальные

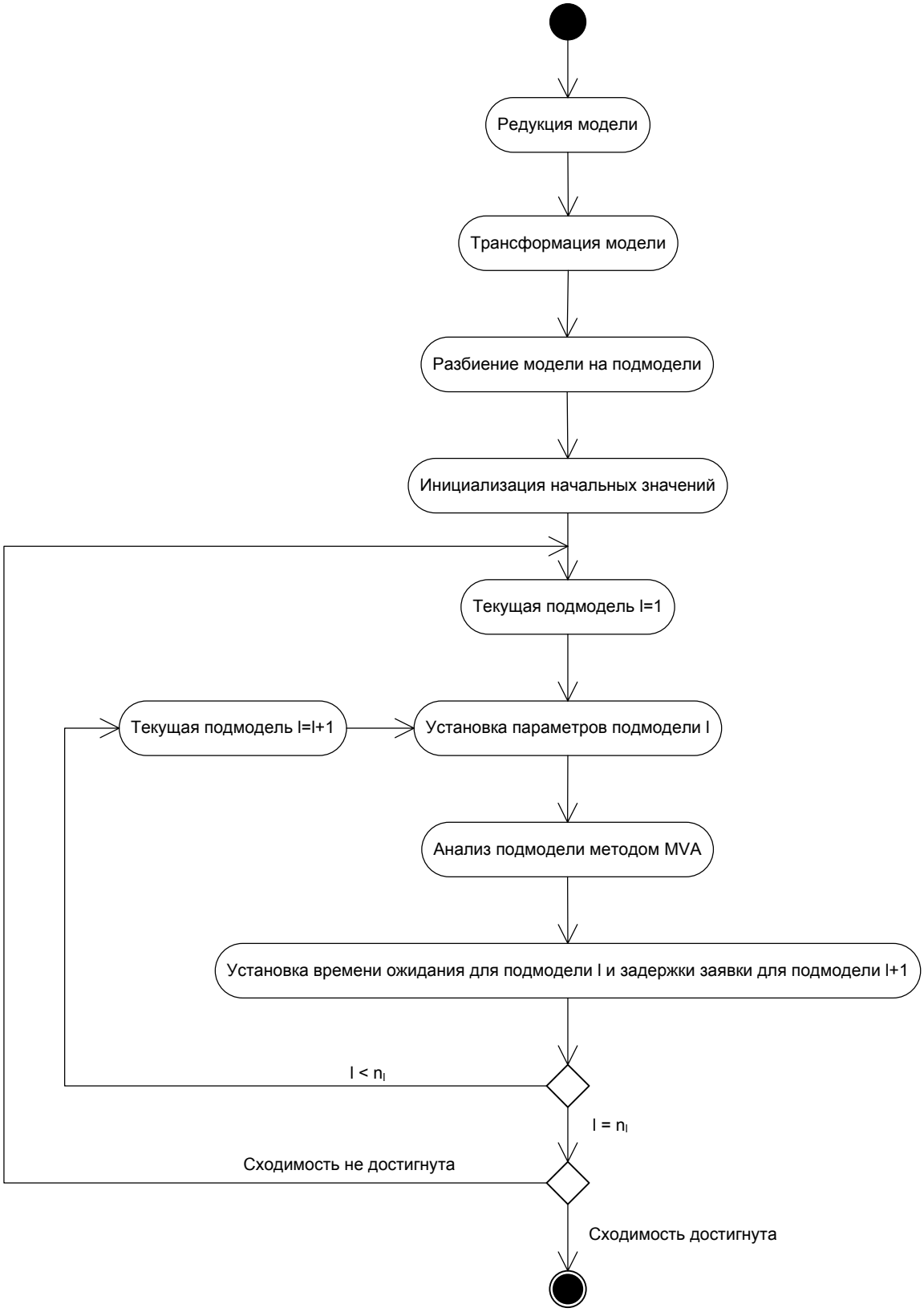

Рис. 3. Блок-схема алгоритма анализа модели производительности в виде диаграммы деятельности UML.

параметры, такие как задержки потоков заявок во всех сетях, кроме первой и время обслуживания для приборов во всех сетях, кроме последней, меняются в процессе анализа сетей.

Активные сервера, как принимающие, так и осуществляющие вызовы, в подмоделях играют роль как клиентов, так и серверов.

Если задача m является клиентом в подмодели l, ей соответствует центр обслуживания с задержкой. Его время обслуживания расчитывается как сумма полного времени ожидания w_{mj} к другим приборам обслуживания во всех подмоделях кроме текущей, где этой задаче соответствует поток заявок:

$$s_{ml} = \sum_{i \in L, i \neq l} \sum_{j \in S_i} w_{mj}$$

Кроме того, клиент подмодели отображается также на поток заявок соответствующей сети массового обслуживания. Задержка Z_{il} потока заявок i в подмодели l определяется на основе коэффициента использования $p_{i,l-1}$ и пропускной способности $\lambda_{i,l-1}$ задачи i, когда она играет роль сервера в подмодели l-1:

$$Z_{il} = N_{il} \frac{1 - p_{i,l-1}}{\lambda_{i,l-1}}$$

Если задача m соответствует серверу в подмодели l, его время обслуживания расчитывается как сумма полного времени ожидания всех потоков заявок, соответствующих этой задаче, во всех подмоделях:

$$s_{ml} = \sum_{i \in L} \sum_{j \in S_i} w_{mj}$$

Алгоритм анализа модели предполагает поочередный анализ подмоделей начиная с первой и заканчивая последней. Результаты анализа очередной подмодели фиксируются в ней, а также отражаются в следующей подмодели.

Внешний цикл алгоритма останавливается по достижению критерия сходимости. Таким критерием является допустимая разница между временем обслуживания прибора на итерации i и $i+1$. Если средняя разница по всем приборам не превышает допустимую, работа алгоритма завершается.

Основным результатом, получаемым в результате анализа сетей массового обслуживания методом анализа среднего значения, является общее время ожидания, показывающее среднее время, затрачиваемое заявкой определенного типа на обработку с помощью конкретного прибора обслуживания. Общее время ожидания включает не только собственно время обслуживания прибором, но и время ожидания в очереди.

Производными показателями являются:

- общее время ожидания заявки, складывающееся из суммы времени ее ожидания на каждом приборе обслуживания;
- пропускная способность потока заявок, определяемая как отношение численности потока к сумме общего времени ожидания и задержке.

Результаты анализа первой подмодели позволяют выявить метрики производительности с точки зрения пользователя. Общее время ожидания для потока заявок, соответствующего начальной задаче (и, как следствие, актеру исходной модели UML), отражает общее время отклика во время всех обращений пользователя к системе, предусмотренных моделью производительности.

Если при редукции модели производительности задачи, соответствующие вариантам использования, не были удалены, можно получить время отклика для каждого варианта использования. Его можно получить из общего времени ожидания потока заявок, соответствующего актеру, на обработку с помощью прибора, соответствующего варианту использования.

Начиная со второй подмодели полученные результаты позволяют определить среднее время выполнения конкретной операции конкретного класса. Для этого необходимо воспользоваться значением общего времени ожидания потока заявок, соответствующего вызывающей задаче, на обработку с помощью прибора, соответствующего вызываемой задаче.

Таким образом, полученные метрики позволяют как проанализировать производительность системы с точки зрения пользователя, так и определить узкие места в ее архитектуре.

Далее в главе рассмотрен вопрос о месте мероприятий по анализу производительности в жизненном цикле разработки ПО при использовании наиболее распространенных методологий (модель водопада, итеративные и гибкие методологии).

В третьей главе рассмотрено разработанное программное средство поддержки информационной технологии анализа производительности в процессе разработки ПО, позволяющее генерировать и анализировать модель производительности.

Программное решение состоит из двух основных модулей:

- бизнес-логики, реализующей функциональность, необходимую для подержки разработанной информационной технологии;
- графического пользовательского интерфейса.

Выделение бизнес-логики как отдельного структурного элемента системы позволяет интегрировать ее функциональность в другие программные приложения, такие как средства моделирования, конфигурационного управления, автоматизированной генерации кода и т.д., тем самым сокращая количество используемых при разработке разнородных программных средств и не требуя их дополнительного изучения или адаптации к ним.

Программное решение реализовано на языке Java с использованием библиотеки SWING для создания графического пользовательского интерфейса. Исходный код системы состоит из 81 класса, общий объем кода которых превышает 5000 строк. Вклад автора диссертации в прооектирование и разработку программного средства составил 100%.

Реализованное программное решение является переносимым и может использоваться на любой программно-аппаратной платформе, поддерживаемой Java SDK 5.0.

В качестве формата исходных данных разработанное приложение использует стандарт XMI (XML Metadata Interchange), предназначенный для обмена объектными моделями, описанными с помощью стандарта MOF (Meta Objects Facility). Спецификация XMI включает отображение метамодели UML на метамодель MOF. Более того, наиболее распространенным способом использования XMI является именно обмен моделями UML.

Внешний вид пользовательского интерфейса приложения показан на рис. 4.

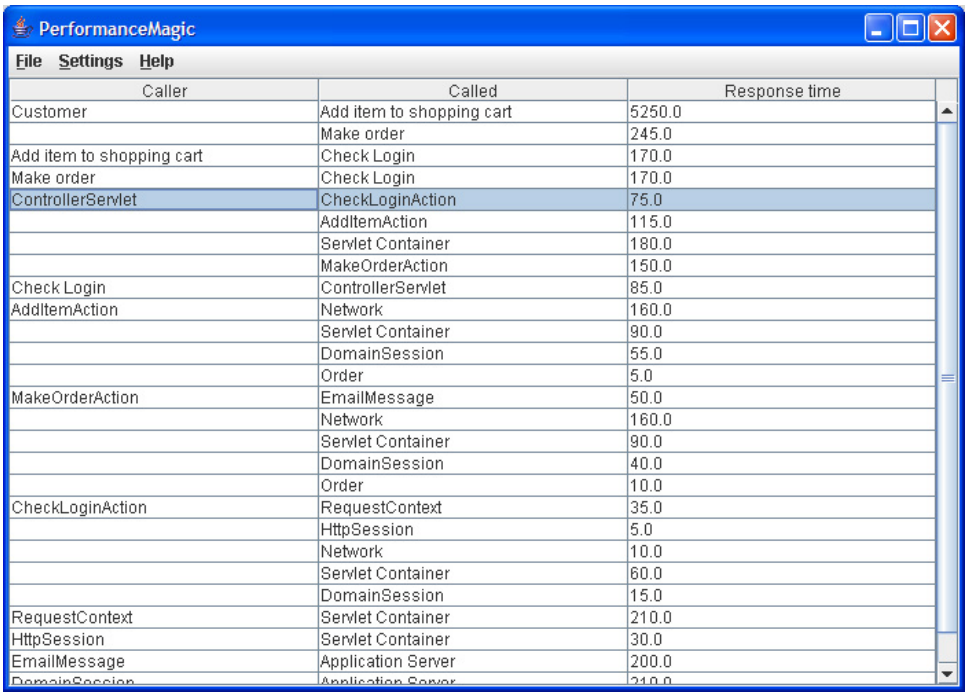

Рис. 4. Внешний вид пользовательского интерфейса разработанного программного решения.

Для того, чтобы оценить эффективность результатов, получаемых с помощью информационной технологии анализа производительности в процессе разработки ПО, было проведено исследование, исходными данными для которого послужило существующее программное приложение (информационный портал), реализованное с использованием UML моделирования. Исследование включало в себя следующие этапы:

- генерация модели производительности на основе UML модели и ее анализ в соответствии с предложенной информационной технологией;

- измерение производительности существующего программного приложения;
- сравнение полученных показателей.

Для того, чтобы наиболее полно оценить эффективность информационной технологии, она была применена для анализа нескольких модулей системы, находящихся на различных архитектурных уровнях:

- уровня веб-сервиса;
- уровня представления HTML;
- уровня бизнес-логики.

Была проведена генерация и анализ модели производительности с последующим измерением на реальной системе следующих вариантов использования:

- вход в систему (Login);
- просмотр объявления (View advertising);
- поиск объявлений (Search advertising).

Модель UML была разработана в CASE-средстве Enterprise Architect, генерация и анализ модели производительности выполнялся с помощью разработанного программного решения.

Сводные результаты исследования представлены в таблице 1.

Таблица 1. Результаты исследования производительности.

Вариант использования	Время отклика (миллисекунды)		
	Результат измерения		Результат анализа
	Доверительный интервал (P=0,95)	Среднее	
Бизнес-логика			
Вход в систему	105,86-108,14	107	110
Просмотр объявления	85,01-86,99	86	85
Поиск объявлений	123,62-126,37	125	120
Представление HTML			
Вход в систему	123,59-126,41	125	130
Просмотр объявления	98,87-101,13	100	105
Поиск объявлений	134,36-137,64	136	140
Веб-сервис			
Вход в систему	115,58-118,42	117	125
Просмотр объявления	103,76-106,24	105	100
Поиск объявлений	130,47-133,53	132	135

Таким образом, результаты исследования показывают, что предложенная информационная технология анализа производительности в процессе разработки ПО является эффективной и позволяет получать адекватные метрики производительности разрабатываемой системы.

В заключении сформулированы основные результаты работы:

1. Определены требования к интеграции анализа производительности в процесс разработки ПО, позволяющей эффективно решать вопросы производительности на протяжении всего жизненного цикла.

2. Проведен анализ существующих технологий анализа производительности в процессе разработки ПО, таких как Software Performance Engineering (SPE), Performance Design & Engineering (PDE), Structure & Performance Specification (SP), Hierarchical Evaluation Tool (HIT). Сделан вывод о том, что в настоящее время подобной информационной технологии, которая бы опиралась на уже существующие артефакты и не требовала бы значительных дополнительных затрат от участников команды, не существует.

3. В рамках анализа требований к информационной технологии анализа производительности в процессе разработки ПО, рассмотрены существующие средства моделирования производительности ПО, такие как сети Петри, сети массового обслуживания, алгебры процессов и имитационное моделирование. Обоснован выбор многоуровневых сетей массового обслуживания для построения и анализа модели производительности в рамках разрабатываемой информационной технологии.

4. Предложена информационная технология анализа производительности в процессе разработки ПО, использующая в качестве исходных данных модель UML. Выбор этой нотации связан с тем, что она является признанным стандартом для моделирования объектно-ориентированных программных систем и уже используется в процессе разработки в большом количестве команд и фирм-производителей.

5. Предложен алгоритм автоматической генерации модели производительности в виде многоуровневой сети массового обслуживания на основе диаграмм UML.

6. Модифицирован алгоритм анализа многоуровневых сетей массового обслуживания для эффективной поддержки моделей, полученных на основе диаграмм UML. Предложены алгоритм редукции модели и правила интерпретации результатов анализа для получения метрик производительности.

7. Разработано программное решение поддержки информационной технологии анализа производительности в процессе разработки ПО. Программное средство позволяет загружать модели UML, аннотированные в соответствии с предложенными в данной работе расширениями, из файлов в формате XMI. Таким образом, он может использоваться практически с любыми средствами моделирования, включая такие широкораспространенные системы как Rational XDE или Together.

8. Проведено исследование, показавшее, что метрики производительности, получаемые с помощью реализованного программного решения, адекватны данным, получаемым при измерении производительности реальной системы.

9. По результатам исследования очерчены пути дальнейшего развития информационной технологии, рассмотренной в диссертации. Прежде всего, необходимо расширить использование выразительных возможностей языка UML,

в частности новой версии UML 2.0, в которой появилась поддержка фрагментов. Фрагменты в частности позволяют отобразить такие структурные блоки программы, как условие или цикл, в диаграмме последовательности. Кроме того, стоит рассмотреть возможность моделирования асинхронных взаимодействий с помощью сообщений (которое применяется например в JMS или MSMQ). Что касается разработанного программного решения, необходимо интегрировать его в инструменты, уже используемые командами разработки, такие как средства моделирования.

Публикации по теме диссертации

1. Дубаков С.А. Моделирование производительности программного обеспечения на основе диаграмм UML / Дубаков С.А., Силич В.А. // Актуальные проблемы информатики и информационных технологий: Материалы международной (VIII Тамбовской межвузовской) научно-практической конференции. - Тамбов: Изд-во ТГУ им. Г.Р. Державина. - с. 52-53.
2. Dubakov S. Performance Evaluation Integration into Object-Oriented Design Process / Dubakov S. // Proc. 8th Korea-Russia International Symposium on Science and Technology KORUS 2004. – Tomsk: Tomsk Politechnic University. 2004. - p. 22-24.
3. Дубаков С.А. Интеграция оценки производительности в процесс объектно-ориентированного проектирования / Дубаков С.А., Силич В.А. // Материалы международной научной конференции «Инновации в науке и образовании-2004». - Калининград: УОП КГТУ, 2004. - с. 184.
4. Дубаков С.А. Автоматическая генерация модели производительности на основе диаграмм UML / Дубаков С.А., Силич В.А. // Материалы 5 всероссийской научно-практической конференции «Системы и средства автоматизации». - Томск: Элеси, 2004. - с. 10.
5. Дубаков С.А. Использование набора диаграмм UML для построения моделей производительности / Дубаков С.А., Силич В.А. // Известия ТПУ. - 2005. - Т.308. - N. 3 - с. 154-159.
6. Дубаков С.А. Формирование модели производительности на основе диаграмм UML / Дубаков С.А., Силич В.А. // Материалы V всероссийской научно-технической конференции «Проблемы информатики в образовании, управлении, экономике и технике». - Пенза: Приволжский дом знаний, 2005. – с. 26-30.

7. Дубаков С.А. Создание региональной информационной системы субъектов и рынка образовательных услуг / Дубаков С.А., Александров А.А., Дубаков А.А., Агранович Б.Л. // Тезисы докладов конференции ассоциации научных и учебных организаций – пользователей сетей передачи данных «RELARN-97». - Нижний Новгород: Ассоциация RELARN, 1997. – с. 15-16.
8. Дубаков С.А. Создание распределенной региональной информационной системы субъектов и рынка образовательных услуг / Дубаков С.А., Александров А.А., Дубаков А.А., Агранович Б.Л. // Тезисы докладов всероссийской научно-методической конференции «Телематика-98». - Санкт-Петербург: Телематика, 1998. – с. 31-34.
9. Дубаков С.А. Создание и направления развития региональной информационной системы «Электронная биржа образовательных услуг» / Дубаков С.А., Александров А.А. // Кибернетика и ВУЗ. – 1999. – вып. 29. – с. 17-20.
10. Дубаков С.А. Безопасность интернет-магазина / Дубаков С.А., Мещеряков Р.В. // Интеллектуальные системы в управлении, конструировании и образовании. – Томск: STT, 2001. – с. 51-56.
11. Dubakov S.A. Using Internet Technologies for Informational Support of School Choice / Dubakov S.A., Alexandrov A.A., Dubakov A.A., Agranovich B.L. // Proceedings of the third Russian-Korean Symposium of Science and Technology “KORUS’99”. – Novosibirsk: Novosibirsk State Technical University, 1999. – p. 17-19.