ПРОГРАММИРОВАНИЕ, 2011, № 1, с. 20–35

ТЕСТИРОВАНИЕ, ВЕРИФИКАЦИЯ И ВАЛИДАЦИЯ ПРОГРАММ

УДК 681.3.06

МОДЕЛИ В НАГРУЗОЧНОМ ТЕСТИРОВАНИИ

© 2011 г. Б. А. Позин, И. В. Галахов
ЗАО ЕС ЛИЗИНГ
E-mail: pozin@ec-leazing, igalakhov@ec-leazing.ru
Поступила в редакцию 06.06.2010 г.

Разработаны метамодели для обеспечения адекватности результатов нагрузочного тестирования и его частей: постановки задачи, исходных данных и анализа результатов эксперимента. Представлены средства для адаптации метамоделей к характеристикам конкретного эксперимента по автоматизированному нагрузочному тестированию и оценки производительности широкого класса систем автоматизации бизнес-процессов.

1. ВВЕДЕНИЕ

В нагрузочном тестировании наиболее активно применяются модели при генерации большого количества виртуальных клиентов для тестирования Web-систем и серверов этих систем с целью определения их нагрузочной способности [1]. Моделированию при этом подвергается структура переходов состояний сервисов, которые предоставляет Web-система с учетом вероятностей реализации сервисов и их последовательностей. Имеются также работы по построению моделей для анализа сетей при их нагрузочном тестировании [2]. Такая достаточно универсальная модель, например, позволяет объединить различные инструментальные средства тестирования в единую технологию на основе нескольких типов, так называемых Universal Probe (UP), описывающих архитектуру, политику взаимодействия элементов системы и ресурсных ограничений.

Однако при практическом тестировании ответственных ИС описанные модели не покрывают потребностей планирования нагрузочного тестирования и оценки адекватности полученных результатов.

В данной работе подход, основанный на моделях, рассматривается применительно к нагрузочному тестированию информационных систем (ИС), в рамках которого осуществляется тестирование прикладного программного обеспечения (ПО) в IT-среде функционирования, то есть в среде, включающей СУБД, базы данных, системное ПО, развернутые на оборудовании комплекса технических средств информационной системы.

Обеспечение адекватности постановки задачи, исходных данных и результатов нагрузочного тестирования требует достижения между заказчиком и подрядчиком обоюдного понимания ключевых аспектов планируемого эксперимента. В условиях быстро развивающегося бизнеса это понимание должно достигаться в кратчайшие сроки.

Ключевыми аспектами планируемого эксперимента являются:

– постановка задачи, определяющая цели эксперимента; должна быть увязана с требованиями к системе;

– исходные данные, определяющие объект тестирования и необходимый характер его нагрузки; должны быть определены;

– набор необходимых характеристик и показателей, на основании которых определяются результаты тестирования; должен быть сформирован до начала эксперимента.

При формализации предметной области (для класса систем) формируются несколько фреймов метапонятий - метамоделей, описывающих возможные понятия, которые могут быть существенными при последующем нагрузочном тестировании и оценке адекватности его результатов.

В метамоделях должны быть определены способы сбора исходных данных для подготовки нагрузочного эксперимента, обеспечивающие его адекватность реальному функционированию системы при ожидаемой нагрузке.

Такими исходными данными являются:

– информация о выбранном виде нагрузочного тестирования (оценочное, аналитическое, настроечное, регрессионное);

– информация об измеряемых характеристиках и показателях производительности;

– информация о структуре системы с точки зрения вариантов подачи нагрузки и способов измерений;

– информация о планируемой загрузке в структурированном виде.

Разработаны четыре метамодели, с помощью которых при постановке задачи нагрузочного эксперимента осуществляется выбор необходимых характеристик, показателей и измеряемых величин, адекватно характеризующих процесс функционирования тестируемой информационной системы.

- Метамодель требований – характеризует тип тестируемой системы, состав нефункциональных требований (бизнес-правила и технические требования) тестируемой информационной системы;

- Метамодель системы – описывает структуру системы как сеть систем массового обслуживания (включая состав элементов типа «ресурс»);

- Метамодель нагрузки – представляет собой описание количества и типов требований обслуживания к системе, закон распределения требований обслуживания во времени эксперимента, правила поступления требований обслуживания в систему, точки входа требований обслуживания в систему (логический уровень);

- Метамодель измерений – определяет состав собираемых характеристик, показателей и величин, интерфейс ввода требований в систему, метод их сбора и алгоритмы преобразования, а также критерии оценки полученных результатов.

При планировании нового нагрузочного эксперимента с использованием понятий метамоделей формируются модели требований, системы, нагрузки и измерений путем выбора метапонятий и определения их значений, исходя из свойств тестируемой информационной системы и целей нагрузочного эксперимента. За счет использования метамоделей при планировании нового нагрузочного эксперимента обеспечивается полнота и целостность формируемых моделей.

Для различных видов нагрузочного тестирования и различных систем указанные модели могут отличаться.

В разделе 2 предложены четыре взаимоувязанные метамодели, содержащие описание базовых понятий и заранее известных правил их применения (параметров модели и правил взаимодействия базовых понятий в моделях), дана интерпретация основных характеристик производительности ИС.

В разделе 3 предложена классификация видов нагрузочного тестирования ИС и описана предлагаемая технология автоматизированного нагрузочного тестирования с использованием предложенных моделей и инструментальных средств IBM Rational, а также инструментальных средств собственной разработки. Показано, как с использованием базовых понятий и правил строятся конкретные модели тестируемых ИС при планировании нагрузочного тестирования: модель требований к эксплуатационным характеристикам, модель нагрузки на тестируемую ИС, модель системы и модель измерений, которые описывают все исходные данные для автоматизированного нагрузочного тестирования и предполагаемые результаты. Модели позволяют автоматизировать настройку инструментальных средств автоматизации тестирования на параметры конкретного нагрузочного эксперимента.

В заключении приведены основные практические результаты по опыту применения технологии, основанной на моделях, полученные в течение четырех лет при нагрузочном тестировании банковских систем.

2. МЕТАМОДЕЛИ КАК СРЕДСТВО ОПИСАНИЯ СВОЙСТВ АРТЕФАКТОВ НАГРУЗОЧНОГО ЭКСПЕРИМЕНТА

Метамодели обеспечивают единый подход к постановке задачи как Заказчиком, так и Исполнителем. Они дают такие преимущества как быстрое понимание и согласование целей эксперимента, быстрая разработка путей достижения этих целей и единое понимание способа их достижения. Метамодели являются мостом, соединяющим неформализованные требования Заказчика с формализованным описанием нагрузочного эксперимента в виде моделей. Это позволяет существенно упростить планирование и автоматизацию выполнения нагрузочного эксперимента.

2.1. Метамодель требований

Метамодель требований содержит правила формализации требований к эксплуатационным характеристикам системы. Такие требования не содержат информации по функциям, выполняемым системой, и поэтому называются нефункциональными.

Нефункциональными требованиями могут быть регламенты, определяемые бизнес-правилами организации, в которой эксплуатируется система. Регламенты могут задавать временные рамки выполнения тех или иных процессов в системе. Возможности системы по соблюдению этих временных ограничений напрямую связаны с ее пропускной способностью.

В зависимости от заданных нефункциональных требований формируются цели нагрузочного эксперимента, выбираются исследуемые характеристики.

Для описания нефункциональных требований к системе предназначена метамодель требований. Метамодель требований может быть представлена в следующем виде:
$R = B \cup T$, где
R – множество требований к системе;
B – множество бизнес-правил;
T – множество технических требований.

Бизнес-правила включают или связаны с технологическими процессами, корпоративными регламентами, политиками, стандартами, законодательными актами, внутрикорпоративными инициативами, учетными практиками, алгоритмами вычислений и т.д.

Технические требования устанавливают технические свойства, которыми должна обладать система, например, характеристики производительности, надежности и доступности.

Метамодель требований определяет правила описания вербальной модели требований, содержащей нефункциональные требования системы.

При выборе конкретных значений понятий или конкретных правил фактически, по метамодели, формируется модель конкретных требований тестируемой системы.

2.2. Метамодель системы

Метамодель системы позволяет описать структуру системы как сеть систем массового обслуживания, состоящих из элементов типа «ресурс» и связей между ними.

Метамодель системы имеет сложную структуру и определяет правила формирования модели объекта тестирования, который описывается до уровня привлекаемых к испытаниям устройств и программных комплексов (компонент, сервисов) с определенными характеристиками производительности.

Для метамодели предполагается, что системы определенного класса могут быть представлены некоторой совокупностью понятий и правил их взаимосвязи.

При выборе конкретных значений понятий или конкретных правил, фактически по метамодели, формируется модель конкретной тестируемой системы.

Метамодель системы может быть представлена в следующем виде:
$\sigma = \left\{\{U(p)\}, \{S\}, K_S, K_U\right\}$, где
$\{U(p)\}$ – множество устройств объекта тестирования с характеристиками производительности p;
$\{S\}$ – множество программных комплексов (компонент, сервисов);
K_S – матрица связей между программными комплексами, строками которой являются источники, столбцами – приемники, а в ячейках указывается наличие связи между ними;
K_U – матрица связей программных комплексов с устройствами, характеризующая количество выделенных устройством ресурсов для программного комплекса. Строками этой матрицы

являются программные комплексы, столбцами – вычислительные комплексы. Элементами матрицы являются векторы выделенных ресурсов.

Точками подачи нагрузки и точками сбора характеристик производительности могут быть выбраны любые связи в матрицах K_S и K_U в соответствии с моделью нагрузки и моделью измерений.

Правила описания объекта тестирования достаточно сложны и разделяются на описание программных и аппаратных средств, а также связи между ними.

Уровень детализации системы, как объекта тестирования, определяется целями эксперимента. Это может быть как один программный комплекс (система в целом), так и множество программных комплексов или приложений (модулей).

Описание программных средств основывается на принципе, что тестируемая система рассматривается как черный ящик со множеством входов и выходов. В зависимости от потребностей система может быть декомпозирована на программные комплексы, которые также рассматриваются как черный ящик со множеством входов и выходов, которые могут быть связаны друг с другом. Связи между определенными блоками могут быть как точками подачи нагрузки, так и точками сбора характеристик.

Описание аппаратных средств должно включать в себя статические характеристики оборудования, такие как количество процессоров, объем памяти, и динамические характеристики, такие как правила динамического перераспределения ресурсов. Такие правила определяют возможности совместного использования тех или иных ресурсов, приоритеты такого использования ресурсов тем или иным программным комплексом, весовые коэффициенты предоставления программным комплексам соответствующих ресурсов.

2.3. Метамодель нагрузки

Метамодель нагрузки определяет структуру входного потока нагрузки.

Метамодель нагрузки может быть представлена в следующем виде:
$L = \{F, M, I\}$, где

F – множество функций, характеризующих распределения нагрузки, вводимой в систему;
M – многомерная матрица, размерностями которой могут являться виды нагрузки: источники потоков нагрузки, наименования и типы потоков, а элементами – их количественные характеристики;
I – множество интерфейсов для ввода нагрузки (в виде ссылки на модель системы).

Поток нагрузки структурируется по его динамическим (F) и статическим (M) свойствам.

К *динамическим свойствам* потока нагрузки относятся законы его распределения во времени. Законы распределения могут быть:

– детерминированные;

– вероятностные.

Детерминированные законы распределения нагрузки во времени предполагают наличие расписания поступления заданного объема нагрузки в систему. В предельном случае расписание может содержать время поступления каждой заявки в систему. Вероятностные законы распределения нагрузки во времени предполагают указание нормального, равномерного, экспоненциального или другого закона распределения.

Количественный состав потока нагрузки определяет его *статические свойства.* Количественный состав потока нагрузки может быть структурирован по нескольким критериям, в том числе:

– по видам нагрузки;

– по типам отправителей.

Для рассматриваемого класса систем существует три основных вида нагрузки, а именно:

– трафик;

– сообщения;

– события.

Нагрузка в виде трафика создается пользователями в системе, построенной по архитектуре клиент-сервер. Наиболее часто встречающимися видами трафика являются HTTP- и SQL-трафик. Действия пользователей на автоматизированных

рабочих местах (АРМ) приводят к формированию запросов (например, SQL-запросов) к серверу. В каждой системе присутствуют различные типы АРМов – отправителей запросов.

Другим видом нагрузки является нагрузка в виде сообщений. Сообщения используются в качестве единицы обмена информацией между частями больших систем. В зависимости от назначения сообщения могут быть разделены на виды. Сообщения, содержащие документы для обработки, в свою очередь могут подразделяться на типы в соответствии с их форматами. Кроме того, сообщения могут быть:

– одиночными (содержащими один документ);

– пакетами (содержащими набор документов).

В зависимости от отправителя, каждое сообщение может шифроваться и подписываться одной или несколькими электронными цифровыми подписями.

Существует нагрузка, создаваемая самой тестируемой системой в результате наступления тех или иных событий. К ним относится выполнение тех или иных регламентных процедур по расписанию или запускаемых оператором.

При выборе конкретных значений понятий или конкретных правил по метамодели формируется модель нагрузки конкретной тестируемой системы. Модель нагрузки описывается на стадии планирования эксперимента.

2.4. Метамодель измерений

Метамодель измерений предназначена для унификации описания:

– способов получения измеряемых величин при нагрузочном тестировании ИС;

– принципиальных возможностей постановки процесса измерений;

– типовых способов оценки показателей и характеристик;

– общих свойств инструментальных средств для анализа возможностей их использования для автоматизации измерений.

Метамодель измерений может быть представлена в следующем виде:

$\Delta = \left\{ \{|U|\}, \tau, \mu, R, \omega \right\}$, где

$\{|U|\}$ – перечень измеряемых величин для каждого типа устройств информационной системы или ее части;

τ – периодичность и скважность измерений;

μ – множество расчетных показателей и их взаимосвязь с измеряемыми величинами;

R – типовые правила и алгоритмы получения расчетных показателей;

ω – типовые критерии оценки полученных результатов.

При подготовке нагрузочного эксперимента осуществляется выбор конкретных значений понятий или конкретных правил, то есть фактически по метамодели формируется модель конкретных измерений.

Для планирования нагрузочного эксперимента ключевую роль играет выбор перечня измеряемых характеристик производительности, так как на основе полученных значений этих характеристик делаются выводы о результатах эксперимента.

Основными видами характеристик производительности являются реактивность, продуктивность и использование (см. таблицу 1).

Они, в свою очередь, разбиваются на несколько расчетных (или непосредственно измеряемых) показателей, значения каждого из которых может рассчитываться на основе непосредственно измеряемых величин, состав которых и их взаимосвязь при расчете показателей могут зависеть от системотехнической платформы и структуры тестируемой системы.

2.4.1. Реактивность

Реактивность важна для систем, работающих в режиме реального времени, то есть требующих выполнения ограничений на время решения определенных или всех функциональных задач. В этом случае эксплуатационными характеристиками ИС могут быть время отклика (реакции) системы на запрос пользователя или время ожидания решения задачи (например, может решаться задача минимизации времени ожидания выполнения бизнес-транзакции).

Таблица 1. Виды характеристик производительности

Характеристики производительности	Расчетные показатели
Реактивность	Среднее время отклика
	Среднее время ожидания
	Среднее время обслуживания
Продуктивность	Пропускная способность
	Выработка
Использование	Утилизация ресурса
	Относительная пропускная способность

Характеристики реактивности определяются как время между предъявлением системе входных данных и появлением соответствующих выходных данных. Реактивность может измеряться как с точки зрения конечного пользователя, так и с точки зрения обрабатывающего центра. Измерению подлежат как бизнес-транзакции, так и физические (технические) транзакции. Транзакция – это единица работы, предназначенной для решения бизнес-задач. Например, банковская транзакция может состоять из операции двойного ввода платежного поручения, то есть заказчика интересует время выполнения не только однократного ввода, но и всей банковской операции. При реализации банковской транзакции может выполняться несколько физических транзакций, например, обращений к базе данных.

Реактивность оценивают или рассчитывают на основе следующих показателей производительности:

– среднее время отклика;

– среднее время обслуживания;

– среднее время ожидания.

Время отклика рассчитывается как период между началом транзакции и завершением выполнения последнего этапа транзакции.

Время отклика (Tr) обычно складывается из времени обслуживания (время, за которое производится обработка) (Ts) и времени ожидания (время ожидания ресурса) (Tw): $Tr = Ts + Tw$.

Показатели реактивности зависят от типа системы, структуры ее точек входа и выхода.

2.4.2. Продуктивность

Для значительного количества систем важна их интегральная пропускная способность, оцениваемая как количество бизнес-транзакций, обрабатываемых системой в единицу времени (продуктивность).

При оценке продуктивности, как правило, используют непосредственно измеренные или расчетные значения следующих показателей производительности:

– выработка (V);

– пропускная способность (или абсолютная пропускная способность) (C).

Выработка (V) определяется как количество завершенных транзакций за определенный период времени:

$$V_i = \frac{N_i}{T}, \text{ где}$$

$i = \overline{1, n}$ - порядковый номер периода времени;
N_i - количество завершенных транзакций;
T - период времени.

Пропускная способность (C) - максимально возможное количество завершенных транзакций за единицу времени:
$C = \max(V_1, \ldots, V_n)$.

Характеристики продуктивности могут использоваться для оценки как системы в целом, так и ее частей.

2.4.3. Использование

Характеристики использования предназначены для того, чтобы описать, в какой степени используются ресурсы тестируемой системы при заданной нагрузке.

К использованию относятся следующие показатели производительности:

– утилизация ресурса (коэффициент использования ресурса);

– относительная пропускная способность.

Утилизация ресурса показывает, какую часть времени ресурс используется для обслуживания транзакций. Общая формула:

$$U = \frac{\sum_{i=1}^{n} t_{Si}}{T} \times 100\%, \text{где}$$

n - количество обслуженных транзакций;
t_{Si} - время обслуживания i-ой транзакции;
T - период обслуживания.

Характеристики использования ресурсов и их количество существенно зависят от используемого в системе оборудования и входящих в его состав ресурсов: процессоров, оперативной памяти, внешних носитетелей, каналов ввода-вывода и т.п.

2.5. Взаимосвязь моделей

Представленные модели взаимосвязаны и тесно взаимодействуют. Для каждого эксперимента все четыре модели должны быть определены.

Модель требований является инициирующей моделью.

После формирования целей тестирования, можно определить:

– объект тестирования: какую часть системы будем тестировать – модель системы;

– какие характеристики и показатели будем определять и каким критериям они должны соответствовать – модель требований;

– каковы потоки требований к системе со стороны управляемого процесса – модель нагрузки;

– какие параметры надо измерять и в каких точках, чтобы получить необходимые результаты – модель измерений.

Ряд характеристик модели измерений и модели нагрузки можно описать только после завершения описания модели системы. К таким характеристикам, например, относятся описание точек подачи нагрузки и точек сбора характеристик.

3. ИСПОЛЬЗОВАНИЕ МОДЕЛЕЙ В ТЕХНОЛОГИИ НАГРУЗОЧНОГО ТЕСТИРОВАНИЯ

При нагрузочном тестировании информационных систем в процессе их эксплуатации главной целью является не проверка правильности функционирования прикладных программ (предполагается, что функциональное тестирование ПО проведено), а **оценка эксплуатационных характеристик информационной системы**, в составе которой функционирует прикладное ПО. Обычно это так называемые «Показатели назначения», или нефункциональные требования, то есть пропускная способность/производительность ИС, реактивность при решении функциональных задач и ряд других.

Главную сложность при нагрузочном тестировании составляет обеспечение адекватности исходных данных и результатов нагрузочного тестирования информационных систем представлениям заказчика. Дело в том, что заказчик не всегда может точно сформулировать цели тестирования, сведения об особенностях нагрузки ИС и достаточно точно заранее, до проведения нагрузочного тестирования, составить представление о том, какие результаты могут и должны быть получены. В силу этого планирование и проведение нагрузочного эксперимента могут затягиваться на период 2-3 месяца. Из-за недостаточно точной постановки задачи может потребоваться повторное проведение нагрузочных экспериментов, стоимость каждого из которых весьма высока.

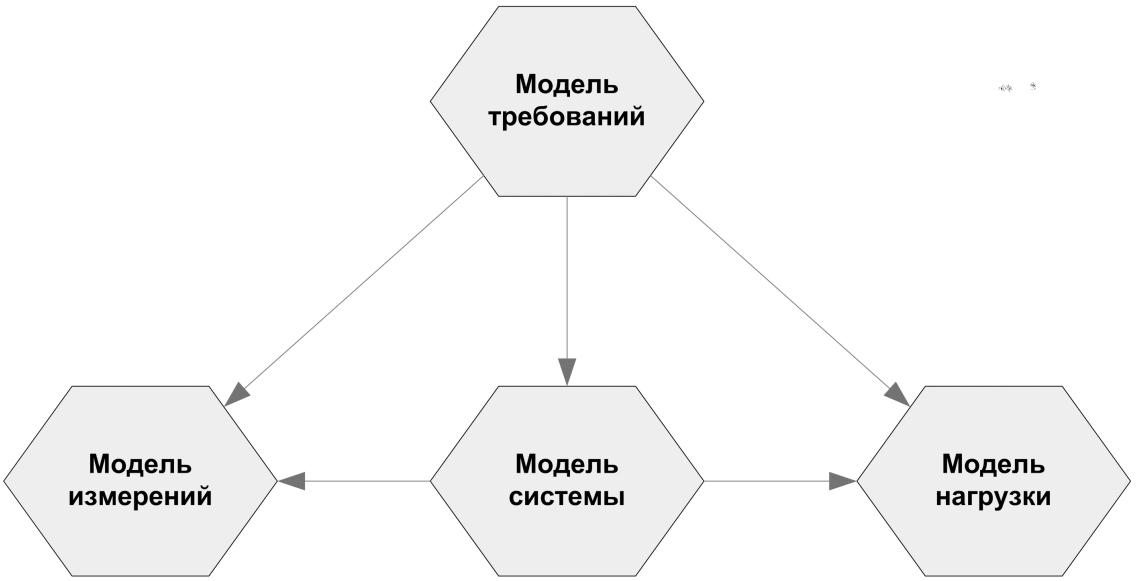

Рис. 1. Взаимосвязь моделей.

Для повышения качества и уменьшения стоимости нагрузочного тестирования необходима комплексная технология автоматизированного нагрузочного тестирования, которая:

1. понятна и заказчику, и тестировщику;
2. обеспечивает возможность документирования:
   - всех сценариев тестирования,
   - предположений о структуре и составе нагрузки,
   - состава и структуры тестируемой системы, как объекта тестирования,
   - принципов и способов измерения характеристик функционирования ИС в точках системы, в которых будут проводиться измерения.

Поэтому актуальной является разработка такой технологии нагрузочного тестирования для достаточно широкого класса систем автоматизации бизнес-процессов, в том числе для банковских систем. Такие системы характеризуются наличием требований по выполнению бизнес-процессов в регламентные сроки. Технические задания на подобные системы содержат набор нефункциональных требований, таких как производительность, количество одновременно работающих конечных пользователей, ограничения на среднее время пребывания транзакций в системе или реактивность и др., то есть в явном виде задают ожидаемые эксплуатационные характеристики (характеристики производительности).

В зависимости от целей нагрузочного тестирования можно выделить следующие его виды:

1. Оценочное – оценка характеристик производительности в однократном нагрузочном эксперименте.
2. Аналитическое – выявление зависимостей (например, производительности от вычислительных ресурсов) в серии нагрузочных экспериментов.
3. Настроечное – настройка и оптимизация нагрузочных характеристик информационной системы или ее составной части.
4. Регрессионное – многократное периодическое нагрузочное тестирование при неизменных условиях для выявления признаков деградации тестируемой системы.

Перечисленные виды тестирования обычно проводятся как полунатурный эксперимент. Каждый вид тестирования имеет особенности планирования (одиночный эксперимент, серия экспериментов, необходимость хранения результатов эксперимента для статистической обработки исторических данных по проведенным экспериментам). При этом схема проведения одиночного эксперимента достаточно стабильна для оценки каждого вида характеристик производительности ИС.

Технология нагрузочного тестирования, основанная на моделях, состоит из следующих этапов:

– определение целей тестирования;

– разработка программы и методики испытаний;

– подготовка к тестированию;

– подача нагрузки;

– сбор данных;

– интерпретация и анализ результатов.

На рисунке 2 представлена технология нагрузочного тестирования с выделением автоматизированных процессов.

На этапе «Определение целей тестирования» с помощью *метамодели требований* происходит определение правил формализации требований к эксплуатационным характеристикам системы и формируется *модель требований*.

По модели требований осуществляется согласование с заказчиком целей нагрузочного тестирования и его сценария.

На этапе «Разработка программы и методики испытаний» из *модели требований* в документ «Программа и методика испытаний» вносят описание сценария, целей нагрузочного тестирования и требований к оцениваемым эксплуатационным характеристикам.

На данном этапе *метамодель нагрузки* используется для определения возможной статической и динамической структуры и состава потока нагрузки. На основе согласования с заказчиком сценариев и состава потоков нагрузки из *метамодели нагрузки* формируется *модель нагрузки*.

Метамодель системы позволяет описать структуру системы в виде *модели системы* как сети систем массового обслуживания. *Модель системы* содержит требования к стенду нагрузочного тестирования, которые согласовываются с заказчиком.

На основе *метамодели измерений* в *модели измерений* документируется набор необходимых измеряемых в ходе эксперимента характеристик и расчетных показателей, способов их получения и критериев оценки результатов тестирования.

На этапе «Подготовка к тестированию» на основе *модели нагрузки* выполняется настройка средств планирования тестирования и генератора тестовых данных.

Тестовые данные генерируются автоматически в соответствии с определенным в *модели нагрузки* количественным составом каждого типа нагрузки. Сгенерированные тестовые данные помещаются в базу тестовых данных.

По *модели системы* осуществляется настройка инструментальных средств проверки стенда и настройка средств автоматизации измерений в части точек сбора информации и используемых механизмов сбора. Настройка средств автоматизации измерений в части перечня собираемых характеристик выполняется по модели измерений.

На этапе «Подача нагрузки» средства автоматизации выполняют процедуры подачи нагрузки в точки входа, заданные для каждого типа нагрузки в *модели системы*. Нагрузка извлекается из заранее подготовленной базы тестовых данных и подается автоматически по расписанию (закону распределения во времени), заданному для каждого типа нагрузки в *модели нагрузки*.

На этапе «Сбор данных» выполняется автоматизированный сбор значений измеряемых характеристик. Точки сбора этих значений средства измерений получают из *модели системы*, а состав измеряемых характеристик – из *модели измерений*.

На этапе «Интерпретация и анализ результатов» средствами автоматизации используются все четыре модели. *Модель требований* используется для сопоставления результатов эксперимента с требованиями к системе. *Модель нагрузки* и *модель системы* обеспечивают формирование отчета об условиях проведения

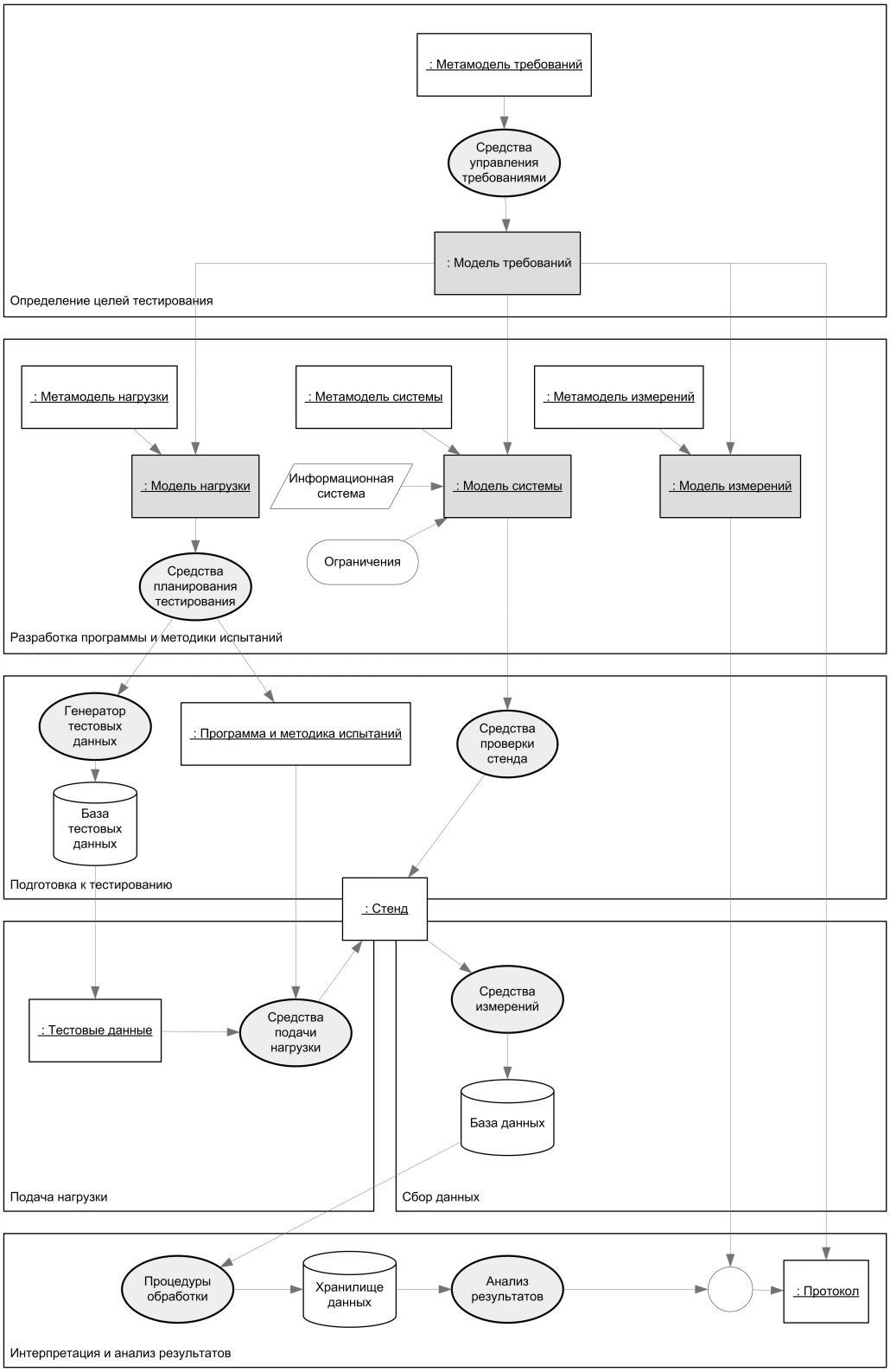

Рис. 2. Процесс нагрузочного тестирования с использованием моделей.

эксперимента. На основе *модели измерений* автоматизировано выполняется сравнение результатов эксперимента с критериями оценки.

Ниже представлено более детальное описание каждого этапа.

3.1. Определение целей тестирования

Цели тестирования определяются исходя из потребностей Заказчика по оценке или прогнозированию эксплуатационных характеристик информационной системы и соответствующих нефункциональных требований.

Заказчик не всегда может полноценно определить цель тестирования. В процессе согласования основной цели, в процессе разработки программы и методики испытаний, могут быть выявлены второстепенные цели тестирования ограничения, которые опять необходимо согласовывать. Разработанная метамодель требований помогает на раннем этапе правильно анкетировать заказчика и полноценно определить цели тестирования, что сокращает количество итераций согласования целей, а также общее время формирования основных и неосновных целей нагрузочного тестирования.

3.1.1.Построение модели требований

На этапе определения целей тестирования с использованием метамодели требований производится анализ объекта тестирования и формируется первичная модель требований, которая в дальнейшем обсуждается с заказчиком, редактируется и утверждается.

Модель требований представляет собой вербальное описание нефункциональных требований, сгруппированных по типам и объектам, к которым они применимы. Эта модель содержит также критерии оценки получаемых в результате нагрузочного тестирования характеристик и расчетных показателей.

Для автоматизации формирования модели требований используется инструментальное средство IBM Rational RequisitePro.

3.2. Разработка программы и методики испытаний

Один из самых важных и сложных этапов нагрузочного тестирования. На данном этапе формируются модель системы, модель нагрузки и модель измерений на основе соответствующих метамоделей.

Три модели формируются параллельно и тесно связаны друг с другом. Например, определение интерфейсов подачи нагрузки и точек сбора характеристик возможно только после описания модели системы.

По завершении данного этапа все согласованные модели включаются в состав документа «Программа и методика испытаний», который представляет собой план нагрузочного тестирования и утверждается заказчиком.

3.2.1. Построение модели системы

На этапе разработки программы и методики согласно поставленным целям разрабатывается вербальная и графическая модель системы, которая в дальнейшем обсуждается с заказчиком, редактируется и утверждается.

Модель системы представляет собой описание объекта тестирования, в которое входят программные и технические средства, их связи и выделяемые ресурсы (память, процессор и т.п.).

На рисунке 3 представлен пример модели системы в виде графической схемы взаимодействия программных и вычислительных комплексов.

Матрицы K_S и K_U для представленной схемы выглядят следующим образом:

$$K_S = \begin{pmatrix} 0 & 1 & 0 & 0 & 0 \\ 1 & 0 & 1 & 1 & 0 \\ 0 & 1 & 0 & 1 & 0 \\ 0 & 1 & 1 & 0 & 1 \\ 0 & 0 & 0 & 1 & 0 \end{pmatrix};$$

$$K_U = \begin{pmatrix} p_1 & 0 & 0 & 0 \\ 0 & p_2 & 0 & 0 \\ 0 & 0 & p_3 & 0 \\ 0 & 0 & 0 & p_4 \\ 0 & 0 & 0 & p_4 \end{pmatrix}, \text{где } p_i -$$

вектор характеристик производительности, характеризующий количество выделенных на вычислительном комплексе ресурсов для заданного программного комплекса.

3.2.2. Построение модели нагрузки

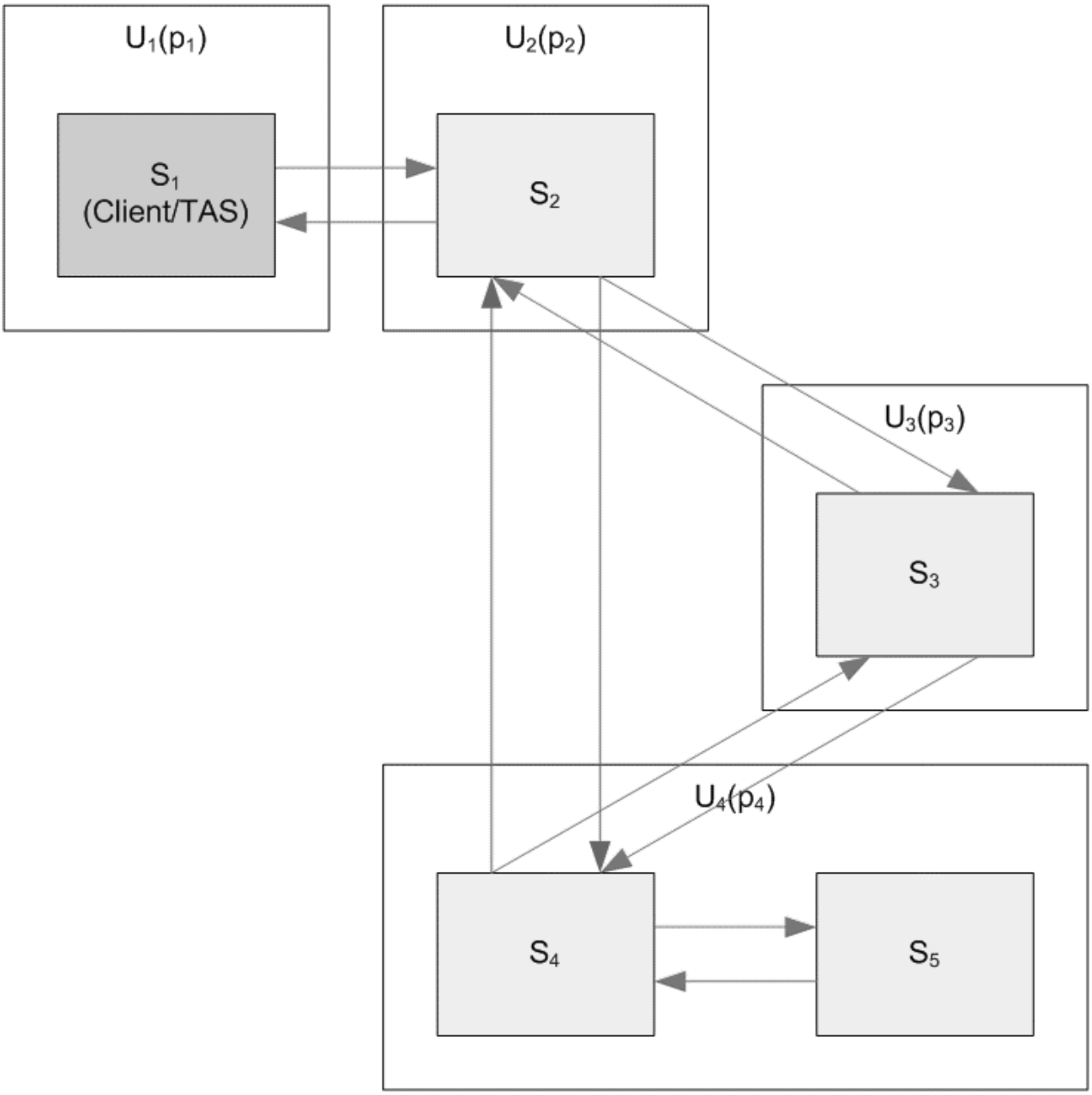

Рис. 3. Пример модели системы в виде графической схемы.

На этапе разработки программы и методики тестирования, согласно поставленным целям, разрабатывается модель нагрузки, которая в дальнейшем обсуждается с заказчиком, редактируется и утверждается. В модели должны быть учтены (как правило, предельные) значения нагрузки на ИС, прогнозируемые заказчиком, а также состав источников нагрузки, который будет имитироваться на период тестирования. Фактически совместно с заказчиком разрабатывается сценарий нагрузки системы, соответствующий целям нагрузочного тестирования. В результате детальной проработки сценария с использованием метамодели нагрузки строится модель нагрузки, в которой все параметры нагрузки определены.

Модель нагрузки представляет собой описание тестовых данных и правил их поступления в систему. Фактически по модели нагрузки формируются исходные данные для генерации нагрузки

инструментальными средствами.

3.2.3. Построение модели измерений

На этапе разработки программы и методики согласно поставленным целям разрабатывается вербальная модель измерений, которая в дальнейшем обсуждается с заказчиком, редактируется и утверждается.

Модель измерений представляет собой описание собираемых характеристик, методов их сбора и интерпретации.

В модели измерений определяется метод и средства сбора значений измеряемых величин. Очень удобно при этом пользоваться метамоделью измерений, в которой определены стандартные механизмы измерений, которые обычно поддерживаются инструментальными средствами, доступными на рынке. Методы сбора значений измеряемых величин могут сильно отличаться в зависимости от поставленных задач тестирования.

По-крупному эти методы можно подразделить на методы неразрушающего контроля, методы разрушающего контроля и смешанные. При неразрушающем контроле для сбора значений измеряемых величин используются системные средства операционного окружения ИС и инструментальные средства автоматизированного тестирования. При разрушающем контроле часть средств измерения встраивают в состав прикладного ПО ИС. При смешанном контроле возможно встраивание средств измерения, как в системные средства, так и в прикладное ПО ИС.

В идеальном случае средства сбора значений измеряемых величин не должны оказывать влияния на тестируемую систему. В случаях, когда необходимо воспользоваться средствами измерений, которые оказывают влияние на тестируемую систему (установка прерываний или триггерных событий), проводятся отдельные испытания с целью оценки степени влияния средств измерения на производительность объекта тестирования. Полученные оценки степени влияния в дальнейшем учитываются при интерпретации результатов.

3.3. Подготовка к тестированию

Для проведения нагрузочного тестирования требуется предварительное выполнение следующих работ:

– подготовка стенда;

– подготовка тестовых данных;

– подготовка средств подачи нагрузки;

– подготовка средств измерения.

Подготовка тестового стенда осуществляется согласно утвержденной модели системы и включает в себя организацию технических и программных средств и обеспечение выполнения дополнительных ограничений, например, синхронизация времени на всех установках объекта тестирования и т.п.

Для этапа подготовки тестирования автоматизирована проверка тестового стенда, основанная на модели системы. Ошибки при подготовке стенда могут привести к срыву нагрузочного эксперимента, так как неработоспособность частей стенда может выявиться уже после начала испытаний. Кроме того, неверные настройки стенда могут исказить результаты нагрузочного тестирования, что может привести к необходимости повторения эксперимента. Наличие формализованной модели системы позволило автоматизировать проверку стенда, сократив время такой проверки. При обнаружении ошибок в конфигурации стенда, настройках его аппаратных и программных комплексов, автоматизированная проверка может повторяться многократно при невысокой трудоемкости за счет автоматизации этого процесса.

Тестовые данные для нагрузочного испытания подготавливаются согласно утвержденной модели нагрузки. Тестовые данные либо генерируются, либо используются реальные данные, подходящие по характеристикам согласно модели нагрузки.

На этапе подготовки тестирования автоматизировано планирование нагрузочного эксперимента и генерация тестовых данных на основе модели нагрузки. При планировании тестирования определяются временные и количественные параметры потока тестовых данных, а также сценарии подачи нагрузки. Средство автоматизации планирования нагрузочного эксперимента

позволяет ввести соответствующие исходные данные о предстоящем испытании в заранее определенные экранные формы. На основе этой информации средство автоматизированной генерации подготавливает тестовые данные. Автоматизированное формирование тестовых данных обеспечивает их полное соответствие текущему состоянию тестируемой системы. Таким образом гарантируется, что все тестовые данные не будут отвергнуты системой и будут обработаны так же, как и реальные.

По модели системы настраиваются интерфейсы подачи нагрузки (и соответствующие инструментальные средства).

При подготовке средств сбора данных по модели измерений настраиваются поставщики и приемники информации.

Результатом этапа подготовительных работ являются готовый к тестированию тестовый стенд (находящийся в исходном состоянии) и готовые для подачи тестовые данные.

3.4. Подача нагрузки

На этапе подачи нагрузки в тестируемую систему подаются подготовленные тестовые данные согласно законам их распределения, описанным в модели нагрузки.

Этап подачи нагрузки автоматизирован с помощью средств тестирования, которые позволяют подавать различные виды нагрузки в соответствии с законом ее распределения во времени. Нагрузка в виде сообщений подается из заранее подготовленной базы тестовых данных. Нагрузка в виде трафика может подаваться как в виде HTTP-, так и SQL-трафика. Состав трафика генерируется на основе предварительного автоматизированного анализа кода запросов, посылаемых АРМами тестируемой системы при выполнении тех или иных операций пользователями. Нагрузка в виде событий формируется с помощью эмуляции действий оператора инструментальными средствами тестирования IBM Rational Robot.

3.5. Сбор данных

Сбор данных производится с помощью средств измерения. На этапе сбора данных автоматизировано собирается первичная информация о значениях измеряемых показателей, таких как загрузка процессоров, использование памяти, время поступления транзакции в систему, время получения отклика системы и т.д. Все получаемые данные сохраняются в оперативной базе данных для последующей обработки.

В зависимости от методов сбора, определенных в модели измерений, данные могут собираться как в процессе тестирования по мере их поступления, так и после прохождения тестирования. Первый случай предпочтителен тем, что собираемые данные по мере поступления можно использовать для контроля прохождения процесса тестирования.

Сбор данных длится до тех пор, пока процесс тестирования не удовлетворит определенным в модели измерений критериям завершения измерений для перехода к следующему этапу.

Первичная обработка данных также зависит от способа обработки, описанного в модели измерений и может выполняться как по мере поступления данных, так и после эксперимента применительно ко всей коллекции собранных данных.

3.6. Интерпретация и анализ результатов

Полученные результаты тестирования интерпретируются согласно модели измерений для дальнейшего анализа: осуществляются расчет показателей и оценка значений эксплуатационных характеристик ИС.

На основе модели измерений производится многошаговая автоматизированная обработка первичных данных. Сначала формируются расчетные показатели на основе заданных в модели измерений правил и алгоритмов. Расчетные показатели помещаются в хранилище данных для последующего анализа. На основе данных хранилища формируются аналитические отчеты, содержащие значения расчетных показателей и критерии их оценки в соответствии с моделью измерений и моделью требований. Благодаря наличию хранилища возможен анализ не только результатов проведенного нагрузочного эксперимента, но и сравнительный анализ нескольких регрессионных тестов.

По завершении интерпретации результатов делаются выводы о соответствии ИС поставленным целям испытания с учетом ограничений и

Таблица 2. Длительность этапов работ

Этап	Длительность	
	до внедрения технологии	в настоящее время
Определение целей тестирования	2 недели	1 неделя
Разработка программы и методики испытаний	5 недель	2-3 дня
Подготовка к тестированию	3 недели	2 дня
Подача нагрузки Сбор данных	в зависимости от плана эксперимента	
Интерпритация и анализ результатов	1 неделя	3-4 часа

критериев модели требований. Результаты испытаний на практике оформляются в виде протокола нагрузочного тестирования.

4. ЗАКЛЮЧЕНИЕ

Принципиально важным вопросом при проведении нагрузочного тестирования является выработка общего понимания постановки задачи заказчиком и исполнителем. Часто достижение такого понимания превращается в длительный и трудоемкий процесс, так как требуется выработка общей для обеих сторон понятийной базы.

Даже при возникновении потребности в нагрузочном тестировании у одного заказчика цели разных экспериментов могут отличаться, поскольку у заказчика возникают разные проблемы, связанные с эксплуатационными характеристиками. Типичными проблемами могут быть по крайней мере следующие:

- Достаточно ли имеющегося оборудования для обеспечения заданной производительности ИС в течение ближайшего периода (1-2 года) при известном ежегодном росте нагрузки?
- Сможет ли ИС, при росте нагрузки обеспечивающая заданную пропускную способность, одновременно обеспечить и реактивность в известных пределах?
- Не появились ли признаки деградации ИС после внесения изменений в программное обеспечение?

и ряд других.

В ходе выполнения достаточно большого количества нагрузочных экспериментов выявлена принципиальная возможность типизировать целый ряд постановок задач нагрузочного тестирования для решения «типичных» задач.

Описанная технология применяется в течение четырех лет. Она использовалась для тестирования нескольких банковских систем на различных платформах: Windows, HPUX, zLinux, z/OS. В различных экспериментах по нагрузочному тестированию на вычислительной установке одновременно функционировало до 40 приложений, 40 баз данных. Максимальный объем базы данных, с которой работали тестируемые приложения, составлял до 2,5 Тб.

Ранее эксперименты такого масштаба в компаниях заказчика требовали большой ручной работы. В случае их проведения для формирования и подачи нагрузки использовалась запись имевшихся потоков нагрузки.

Проведение адекватных испытаний для оценки ожидаемых характеристик ИС в короткие сроки и при относительно небольших затратах трудоемкости (5-10 ч.мес.) было технически невозможно.

Оценка ситуаций использовавшимися метода-

ми была крайне трудоемкой и в большинстве случаев не проводилась. Систематический упреждающий контроль деградации характеристик производительности при внесении изменений в ПО не проводился в силу высокой трудоемкости подобных работ.

Рассмотренная технология позволила выполнить за последние четыре года следующие объемы экспериментальных работ (суммарные данные):

– количество нагрузочных экспериментов – более 40;

– количество сгенерированных сообщений – более 75 млн.;

– среднее количество сообщений в одном эксперименте – около 2 млн.

В таблице 2 представлены данные по средней продолжительности работ до внедрения технологии и в настоящее время.

Приведенные в таблице данные показывают, что разработанная технологии и средства ее автоматизации дают возможность готовить и проводить нагрузочное тестирование в приемлемые для заказчика сроки. Количество повторяемых экспериментов для получения требуемых характеристик снизилось с двух-трех до одного.

При проведении тестирования крупных систем, работающих на дорогих технических средствах важна минимизация таких рисков, как сдвиг сроков испытания, проведение повторных испытаний и т.п. Такие риски возникают, как правило, из-за отсутствия единого понимания заказчиком и подрядчиком целей тестирования и ошибок планирования эксперимента.

Применяя разработанную технологию, удалось уменьшить такие риски и снизить длительность и трудоемкость нагрузочных испытаний, а следовательно и их стоимость.

В работе представлена технология организации и проведения автоматизированного нагрузочного тестирования для класса информационных систем. Технология основана на использовании метамоделей, описывающих требования к проведению нагрузочного эксперимента и свойства объекта тестирования и нагрузки на него. В рамках технологии поэтапно с использованием метамоделей строятся модели, которые понятны заказчику и по которым осуществляется согласование с ним основных параметров эксперимента и критериев оценки результата. С помощью моделей настраиваются инструментальные средства, после чего эксперимент проводится автоматически под управлением инструментальных средств нагрузочного тестирования. Технология охватывает все аспекты планирования, проведения нагрузочного эксперимента и анализа его результатов. Использование моделей существенно (в несколько раз) снижает трудоемкость нагрузочного тестирования по сравнению с имевшимся опытом и обеспечивает возможность повторного использования подготовленного эксперимента, например, при контроле деградации информационной системы в ее жизненном цикле.

СПИСОК ЛИТЕРАТУРЫ

1. *Kim G.-B.* F method of generating massive virtual clients and model-based performance test/ Fifth International Conference on Quality Software, 2005. P. 250-254.

2. *Changyou Xing, Guomin Zhang, Ming Chen.* Research on universal network performance testing model/ International Symposium on Communications and Information Technologies, 2007. P. 780-784.

3. *Kostogryzov A., V.Panov, B.Pozin, V.Sablin.* Mathematical modeling of processes in systems life cycles in compliance with standarts requirements of ISO/IEC 15288 and ISO/IEC 12207, Spincose, Montreal, Canada, 2003.

4. *Козлов А.Н., Позин Б.А.* Технология комплексного нагрузочного тестирования банковской системы, международная научно-практическая конференция «Реинжиниринг бизнес-процессов на основе современных информационных технологий системы управления знаниями», 2006 г. (РБП-СУЗ-2006), Москва. С. 130-131.

5. SO/IEC 1539 "Information technology – Software engineering – Software measurement process".

6. Prof. Flavio Oliveira. Performance Testing: an Introduction FACIN-PUCRS, 2008.

7. ГОСТ 34.602-89. ИНФОРМАЦИОННАЯ ТЕХНОЛОГИЯ. Комплекс стандартов на автоматизированные системы. Техническое задание на создание автоматизированной системы.

ISSN 0361-7688, Programming and Computer Software, 2011, Vol. 37, No. 1, pp. 15–25. © Pleiades Publishing, Ltd., 2011.
Original Russian Text © B.A. Pozin, I.V. Galakhov, 2011, published in Programmirovanie, 2011, Vol. 37, No. 1.

Models in Performance Testing

B. A. Pozin and I. V. Galakhov

ES-Leasing, Varshavskoe shosse 125, Moscow, 117405 Russia
E-mail: bpozin@ec-leasing.ru; igalakhov@ec-leasing.ru

Received July 15, 2010

Abstract—Metamodels ensuring the adequacy of the results of performance testing and its parts were developed; these parts are statement of the problem, initial data, and analysis of experimental results. Tools for the adaptation of metamodels to the characteristics of a specific automated load testing experiment and for the estimation of performance of a wide class of systems for automating business processes are described.

DOI: 10.1134/S036176881101004X

1. INTRODUCTION

In performance testing, the mostly used models are those for generating a large number of virtual clients for testing Web-systems and their servers to determine the load they can sustain [1]. In this case, one simulates the structure of transitions between the states of services provided by a Web-system with account for the probabilities of running these services and their sequences. There are also studies devoted to the analysis of networks in performance testing [2]. Such a versatile model, for example, makes it possible to combine different testing tools into a single technology on the basis of several types of the so-called Universal Probes (UPs), which describe the architecture, the policy of interaction between the system elements, and resource limitations.

However, when testing mission-critical information systems (ISs) in practice, the above-mentioned models do not cover the needs of performance testing planning and assessment of the adequacy of results.

In this study, the model-based approach is applied to performance testing of ISs; within this framework, we test application software in an IT-environment (i.e., in an environment that includes a database management system (DBMS), databases, system software, and the corresponding hardware).

To ensure that the problem statement, its initial data, and the results of performance testing are adequate, it is required that both the client and the contractor have common understanding of the key aspects of the planned experiment. Under the conditions of quick business development, this realization must be reached as fast as possible.

The key aspects of the planned experiment are:

– the problem statement determining the aims of the experiment must match the system requirements;

– the initial data describing the object to be tested and the required properties of its load must be defined;

– the set of necessary characteristics and indicators that are used to assess the testing results must be specified prior to the beginning of the experiment.

When formalizing the domain (for a class of systems), several metaconcept frames (these are metamodels describing the possible concepts that can be important in the subsequent performance testing and assessment of the adequacy of its results).

Metamodels should define how the initial data are collected for performance testing to ensure that the experiment is adequate to the actual operation of the system under an expected load.

These initial data are as follows.

– Information about the type of performance testing (evaluation, analytical, configuration, or regression testing).

– Information about the parameters to be measured and performance indicators.

– Information about the system structure in terms of load-supply variants and measurement methods.

– Information about the planned load in a structured form.

We developed four metamodels that make it possible to choose the characteristics, indicators, and measured quantities that adequately characterize the operation of the IS under test. The metamodels are as follows.

– Metamodel of requirements characterizing the type of system under test, the set of the non-functional requirements (business rules and technical requirements).

– Metamodel of the system describing the structure of the system as a network of queueing systems (including the configuration of resource-type elements).

– Metamodel of load describing the number and types of service requests to the system, the distribution law of the service requests at the time of the experiment, the rules governing the arrival of service

requests, and the points of their entry into the system (logical level).

– Metamodel of measurements determining the composition of the characteristics, indicators, and quantities to be collected, the interface for inputting requirement into the system, the method of their collection and transformation algorithms, and the criteria for assessing the results.

When a new load testing experiment is planned on the basis of the metamodel concept, models of requirements, system, load, and measurements are generated by choosing metaconcepts and determining their values based on the properties of the IS under test and goals of the load experiment. The use of metamodels in planning a new load testing experiment ensures that the resulting models are complete and integral.

For different types of performance testing and different systems, these models may be different.

In Section 2, we propose four interrelated metamodels describing the basic concepts and the prescribed rules of applying these concepts (parameters of the model and the interaction rules between the basic concepts in models). An interpretation of the basic performance characteristics of ISs is also given.

In Section 3, a classification of the main types of IS performance testing is given, and the proposed technology of automated performance testing using the proposed models, IBM Rational tools, and our own tools are described. It is shown how the basic concepts and rules can be used to construct specific models of ISs when the performance testing is planned; these are the model of requirements for the operational characteristics, the model of load, the model of the system, and the model of measurements, which describe all the initial data needed for the automated performance testing and the expected results. The models make it possible to automate the configuration procedure of the automated testing tools for the parameters of a specific load experiment.

The Conclusion gives the main practical results obtained from our four-year long experience of using the model-based technology for testing banking systems.

2. METAMODELS AS A TOOL FOR DESCRIBING THE PROPERTIES OF ARTIFACTS OF LOAD EXPERIMENT

Metamodels provide a unified approach to the problem statement both for the Client and the Contractor. Their benefits include quick understanding and accommodation on the experiment goals, quick construction of ways for achieving these goals, and a unified understanding of the way for their fulfillment. Metamodels are a bridge connecting the Client's nonformalized requirements with a formalized description of the load experiment in the form of models. This can substantially simplify the planning and automation of the load experiment.

2.1. Metamodel of Requirements

The metamodel of requirements contains rules for formalization of requirements for the operational characteristics of the system. These requirements contain no information concerning the functions performed by the system and, therefore, are called nonfunctional requirements.

Nonfunctional requirements may involve regulations determined by the business rules of the organization where the system is operated. The regulations can specify the time framework for performing certain processes in the system. Whether the system can comply with these time limitations directly depends on its capacity.

Depending on the given nonfunctional requirements, the goals of the load experiment are set and characteristics to be examined are chosen.

The nonfunctional requirements for the system are described using the metamodel of requirements, which can be represented in the following form.

$$R = B \cup T, \text{ where}$$

R is the set of requirements for the system,

B is the set of business rules,

and T is the set of technical requirements.

Business rules include or are related to engineering procedures, corporate regulations, policies, standards, legal acts, intra-corporate initiatives, registry practices, computing algorithms, etc.

Technical requirements set technical features of the system, for example, the characteristics of performance, reliability, and accessibility.

The metamodel of requirements sets the rules for describing a verbal model of requirements containing nonfunctional requirements.

When choosing specific values of concepts or individual rules, the model of specific requirements for the tested system is actually generated on the basis of metamodels.

2.2. Metamodel of System

The metamodel of the system makes it possible to describe the system structure as a network of queueing systems consisting of resource-type elements and relations between them.

The metamodel of the system has a complex structure and determines the rules for the generation of the model of the object under test, which is described up to the level of the tools and software packages (components and services) involved in testing with certain performance characteristics.

For the metamodel, it is supposed that systems of a certain class can be represented by a set of concepts and rules of their relationship.

When choosing specific values of concepts or individual rules, the model of specific requirements for the system under test is actually formed on the basis of the metamodel.

The metamodel of the system can be represented as

$$\sigma = \{\{U(p)\}, \{S\}, K_S, K_U\}, \text{ where}$$

$\{U(p)\}$ is the set of tools of the object under test with performance characteristics p,

$\{S\}$ is the set of software packages (components and services),

$\{K_S\}$ is the matrix of relations between software packages in which the rows represent the sources, the columns repesent the recipients, and the cells indicate a link between them,

$\{K_U\}$ is the matrix of relations of the software packages with tools; this matrix indicates the number of resources allocated by a tool for a software package. The rows of this matrix correspond to the software packages and the columns correspond to computer systems. The elements of this matrix are vectors of allocated resources.

Points of load supply and points of collection of performance characteristics can be represented by any links in the matrices $\{K_U\}$ and $\{K_S\}$ according to the model of load and model of measurements.

The rules for describing the object under test are fairly sophisticated; they describe the software, the hardware, and their relationships.

The level of detail of the system considered as an object under test is determined by goals of the experiment. It can be a single software package (the system as a whole) or a set of software packages or applications (modules).

The description of the software is based on the principle that the system under test is regarded as a black box with many inputs and outputs. If needed, the system can be decomposed into software packages, which also are regarded as black boxes with many inputs and outputs, which can be linked one to another. The links between certain blocks can be both points of load supply and points of characteristic collection.

The description of hardware must include static characteristics (such as the number of processors and amount of memory) and dynamical characteristics (such as the rules of dynamic redistribution of resources). These rules determine the possibility for the joint use of some resources, set priorities for the utilization of the resources by one or another software package, and the weight coefficients that regulate the allocation of resources to software packages.

2.3. Metamodel of Load

The metamodel of load determines the structure of the input load flow and can be represented as

$$L = \{F, M, I\}, \text{ where}$$

F is a set of functions characterizing the distribution of the input load,

M is a multidimensional matrix, in which the dimensions can be the load types such as load flow sources, flow names and types, and the elements of the matrix can represent their quantitative characteristics,

I is the set of interfaces for inputting the load (as a reference to the model of the system).

The load flow is structured with respect to its dynamic (F) and static (M) properties.

The dynamic properties of the load flow include the laws of its distribution in time, which can be

– deterministic,

– probabilistic.

Deterministic laws of the load flow time distribution imply a schedule for the arrival of a given amount of load to the system. In the limiting case, a schedule can contain the time of arrival of each demand in the system. Probabilistic laws of the load flow time distribution imply normal, uniform, exponential, or other distribution laws.

The quantitative composition of load flow describes its static properties and can be structured with respect to several criteria, including

– types of load,

– types of senders.

For the given class of systems, there are three main load types:

– traffic,

– messages,

– events.

The traffic load is produced by users in a system based on the client–server architecture. Most frequently used traffic types are the HTTP traffic and the SQL traffic. The actions of users at workstations lead to the generation of queries (for example, SQL queries) to the server. Each system has different types of workstations that can send queries.

Another type of load is the message load. Messages are used as units of information exchange between parts of large systems. Depending on their designation, messages can be divided into types. In turn, the messages containing documents for processing can be classified into types depending on their format. Furthermore, messages can be classified into

– single messages (containing a single document),

– packet messages (containing a set of documents).

Depending on the sender, each message can be encrypted and signed by one or more electronic digital signatures.

There is a load created by the tested system itself as a result of some events (for example, execution of scheduled tasks or the execution of procedures initiated by an operator).

When choosing specific values of concepts or individual rules, the model of specific requirements for the tested system is formed on the basis of the metamodel.

Table 1. Types of performance characteristics

Performance characteristics	Calculated indicators
Reactivity	Average time of response
	Average time of waiting
	Average time of service
Productivity	Capacity
	Output
Utilization	Utilization of resource
	Relative capacity

The load model is described at the stage of planning the experiment.

2.4. Metamodel of Measurements

The metamodel of measurements is designed to unify the description of the following things:

– methods of obtaining measured quantities;

– theoretical possibilities for organizing the measurement procedure;

– typical methods for estimating the indicators and characteristics;

– general features of tools from the viewpoint of their usage for the automation of measurements.

The metamodel of measurements can be represented as

$$\Delta = \{\{|U|\}, \tau, \mu, R, \omega\}, \text{ where}$$

$\{|U|\}$ is the list of measured quantities for each type equipment or the IS part,

τ is the periodicity and off-duty factor of measurements,

μ is the set of calculated indicators and their relationships with the measured quantities,

R describes the typical rules and algorithms for calculating the indicators,

ω describes the typical criteria for estimating the results.

When preparing the load experiment, the specific values of concepts or specific rules are chosen; actually the metamodel is used as a basis for forming the model of specific measurements.

In the planning of the load experiment, the key role is played by the choice of list of measured performance characteristics because these characteristics make it possible to draw conclusions on the experimental results.

The main types of performance characteristics are reactivity, productivity, and utilization (see Table 1).

In turn, they are divided into several calculated (or directly measured) indicators, of which each can be calculated on the basis of directly measured quantities; their composition and relationships in the calculation of the indicators can depend on the systems engineering platform and the structure of the system under test.

2.4.1. Reactivity. Reactivity is important for real-time systems (i.e., when certain or all functions must be performed in a limited amount of time). In this case, the operational characteristics of the IS may include the time of system response (reaction) to the user query or the waiting time for solving a problem (for example, the waiting time for the execution of a business transaction can be minimized).

The reactivity characteristics are determined as the time between the arrival of input data to the system and the appearance of the corresponding output data. The reactivity can be measured in terms of both the end user and processing center. Both business transactions and physical (technical) transactions are to be measured. A transaction is a unit of work designed for executing business tasks. For example, a banking transaction may require ouble entry; i.e., the client cares for the time of execution of the entire banking operation rather than for the time of just inputting the data one time. A banking transaction may involve the execution of several physical transactions (for example, databases accesses).

Reactivity is estimated or calculated on the basis of the following performance indicators:

– average response time;

– average waiting time;

– average time of service.

The response time is calculated as the period between the start of the transaction and the termination of its final stage.

The response time (Tr) is usually composed of the time of service (time of processing) (Ts) and of the waiting time (time of waiting for a resource) (Tw):

$$Tr = Ts + Tw.$$

The reactivity indicators depend on the system type and the structure of its entry and exit points.

2.4.2. Productivity. For many systems, it is important to consider their integral capacity estimated as the number of business transactions processed by the system per unit time (productivity).

The estimation of productivity is normally based on directly measured or calculated values of the following performance indicators:

– output (V)

– capacity (or absolute capacity) (C).

The output (V) is defined as the number of completed transactions per a given time:

$$V_i = \frac{N_i}{T}, \text{ where}$$

$i = \overline{1, n}$ is the serial number of the time period,

N_i is the number of completed transactions,

and T is the time period.

The capacity (C) is the maximum possible number of completed transactions per unit time:

$$C = \max(V_1, ..., V_n).$$

The characteristics of productivity can be used for estimating both the system as a whole and its parts.

2.4.3. Utilization. The utilization characteristics of designed for describing the extent that the resources of the tested system are used under a given load.

Utilization includes the following performance indicators:

– resource utilization (coefficient of resource utilization),

– relative capacity.

The resource utilization shows the fraction of time that the resource is used serving transactions. The general formula is

$$U = \frac{\sum_{i=1}^{n} t_{Si}}{T} \times 100\%, \text{ where}$$

n is the number of transactions served,

t_{Si} is the time needed to serve the ith transaction,

and T is the serving period.

The utilization characteristics and their number considerably depend on the hardware used in the system and its constituent resources: processors, memory, external storage, input–output channels, etc.

2.5. Interrelation of Models

The models discussed above are interrelated and closely interact with one another. For each experiment, all the four models must be determined.

The model of requirements is an initiating model (see Fig. 1).

Upon the testing goals are set, one can determine:

– the object of testing: what part of the system will be tested (*model of the system*);

– which characteristics and indicators must be determined and which criteria should be met by them (*model of requirements*);

– which are the load flows for the system from the controlled process (*model of load*);

– which parameters should be measured and at which points to obtain the necessary results (*model of measurements*).

A number of characteristics of the model of measurements and the model of load can be described only when the description of the model of the system is completed. For example, these characteristics are the description of points where load is applied and the points at which the characteristics are collect.

3. USE OF MODELS IN THE PERFORMANCE TESTING TECHNOLOGY

When performance testing of ISs is performed in the course of their operation, the main goal is to assess the operational characteristics of the IS of which the application software is one of the parts; it is assumed that the functional testing of the software has already been performed earlier). In performance testing, we normally want to determine the so-called **performance characteristics or nonfunctional requirements** (i.e., IS capacity, performance, reactivity in executing functional tasks, etc.).

The main difficulty in performance testing is to ensure that the initial data and results of performance testing of ISs are adequate to the client views. The point is that the client cannot always exactly formulate the testing goals and information about the features of the IS load; nor can the client get a good idea (before the performance testing is completed) of what results can and should be obtained. Due to this, the planning and execution of the load experiment can take as long as 2–3 months. If the task is formulated inaccurately, it may be needed to repeat the load experiments, which is very costly.

To improve the quality and reduce the cost of performance testing, one needs a comprehensive technology of automated performance testing that has the following properties:

– is conceivable to both the client and the tester;

– makes it possible to document

• the test scenarios,

• assumptions about the load structure and composition,

• the structure and composition of the system under test considered as an object of testing,

• principles and ways of measuring the IS operational characteristics at the points where the measurements are to be performed.

Therefore, it is of interest to develop such a technology of performance testing for a wide class of systems for automating business processes, including banking systems. These systems require that business processes be executed in due time. The requirements specifications for such systems contain a set of nonfunctional requirements, such as performance, number of simultaneously operating end users, limitations to the average time of transaction execution or reactivity, etc.; i.e., these tasks explicitly specify the expected operational characteristics (performance characteristics).

Depending on its aims, the performance testing can be of the following types:

– evaluation testing (estimation of performance characteristics in a single load experiment);

– analytical testing (establishing dependences (for example, the dependence of performance on computing resources) in a series of load experiments);

– configuration testing (configuration and optimization of load characteristics of the IS or its part);

– regression testing (multiple periodic load testing under fixed conditions to detect degradation of the system).

These types of testing are usually performed as semi-scale experiments. Each type of testing has spe-

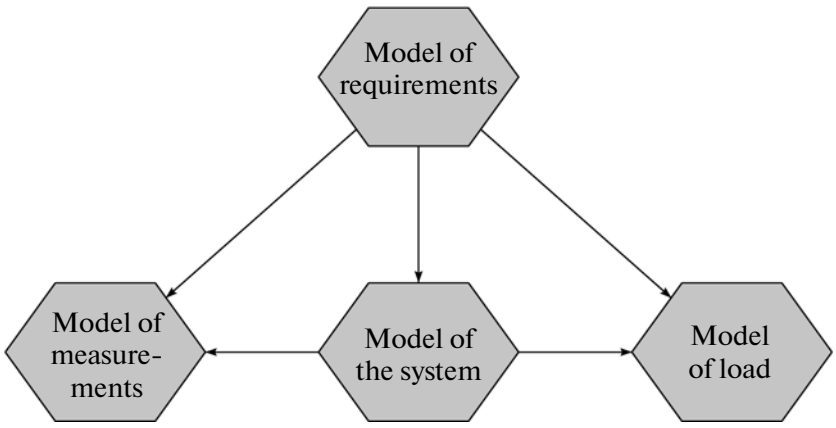

Fig. 1. Interrelation of models.

cific features of planning (single experiment, series of experiments, the requirement to store the experimental results for statistical analysis of historical data obtained from experiments). However, the design of a single experiment is fairly stable for estimating each type of performance characteristics.

The model-based performance testing technology of involves the following phases:

– determination of testing goals;

– development of testing program and procedure;

– preparation for testing;

– load feed;

– data collection;

– interpretation and analysis of results.

Figure 2 shows the performance testing technology (the automated procedures are shaded).

At the goal determination phase, requirements for the operational characteristics of the system are determined using the metamodel of requirements, and a model of requirements is constructed.

Using the model of requirements, the goals of performance testing and its scenario are accommodated with the client.

At the phase of developing the testing program and procedure, the model of requirements is used to bring the description of the scenario, goals of performance testing, and requirements for the operational characteristics to be estimated into the document called *The Program and procedure of testing*.

At this phase, the metamodel of load is used to determine the possible static and dynamic structure and configuration of the load flow. Based on the accommodation of the scenarios and the configuration of the load flows with the client, the model of load is formed based on the metamodel of load.

The metamodel of the system makes it possible to describe the system structure in the form of a model of the system as a network of queueing systems. The model of the system contains requirements for the performance testing workbench that are approved by the client.

Based on the metamodel of measurements, the model of measurements includes a documented set of characteristics measured in the course of the experiment and calculated indicators, the methods for obtaining them, and criteria for estimating the testing results.

At the phase preparation for testing, the model of load is used to adjust the test planning tools and test data generator.

The test data are generated automatically in line with the quantitative configuration of each load type determined in the model of load. The test data generated in this way are placed into the test database.

Using the model of the system, one adjusts the tools for workbench checking and also adjusts the measurement automation tools in what concerns the points of data collection and collection procedures. The adjustment of the measurement automation tools in the part concerning the list of collected characteristics is performed using the model of measurements.

At the load feeding phase, the automation tools feed the load at the input points determined for each load type in the model of the system. The load is retrieved from an earlier prepared test database and applied automatically according to the schedule (time distribution law) specified for each load type in the model of load.

The data collection phase, the values of the measured characteristics are automatically collection. The points of collection are obtained the model of the system, and the configuration of the measured characteristics is obtained from the model of measurements.

At the phase of interpretation and analysis of results, the automation tools use all the four models. The model of requirements is used to compare the experimental results with the system requirements. The model of load and the model of the system generate a report on the experimental conditions. The

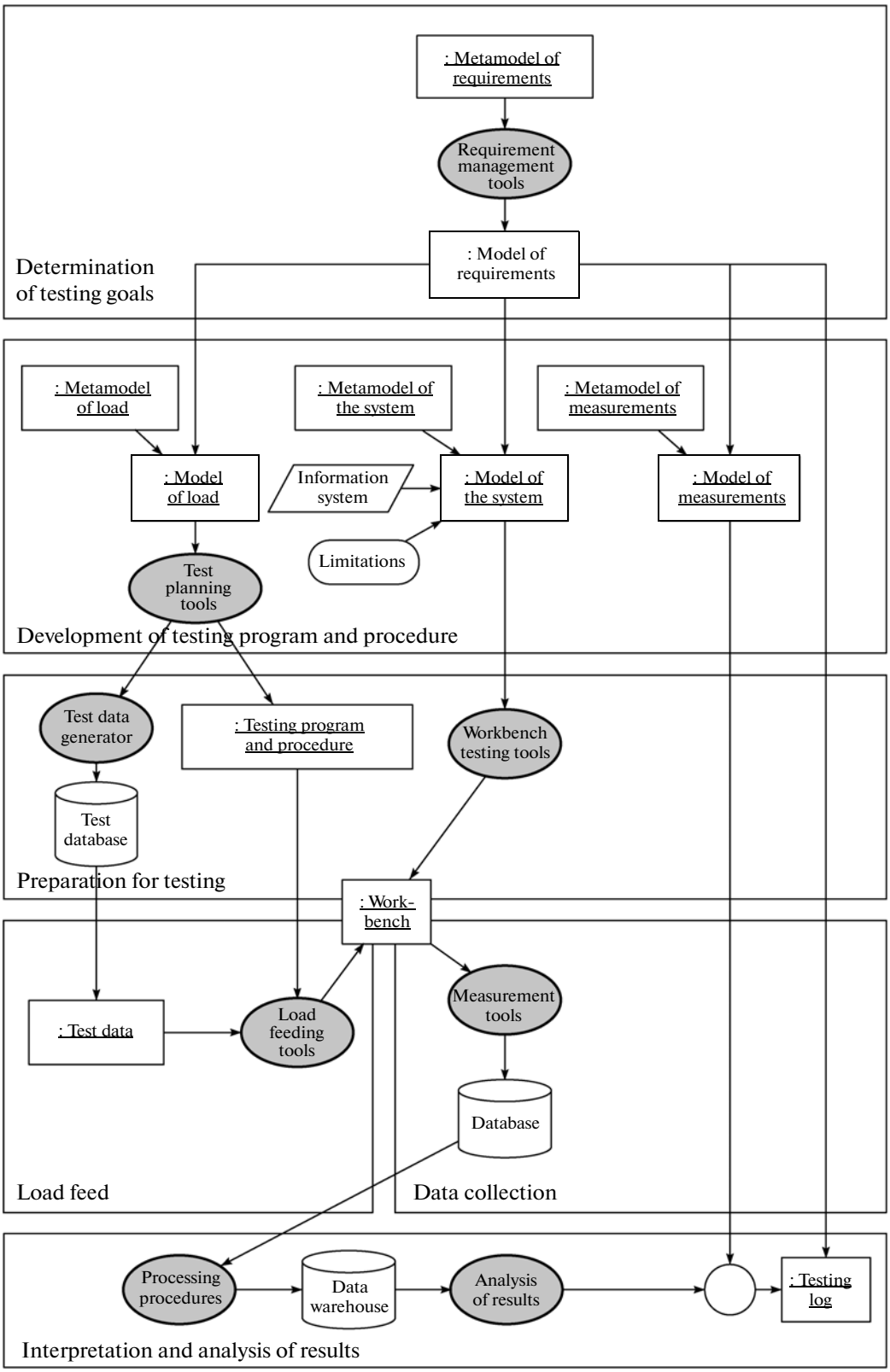

Fig. 2. Model-based performance testing.

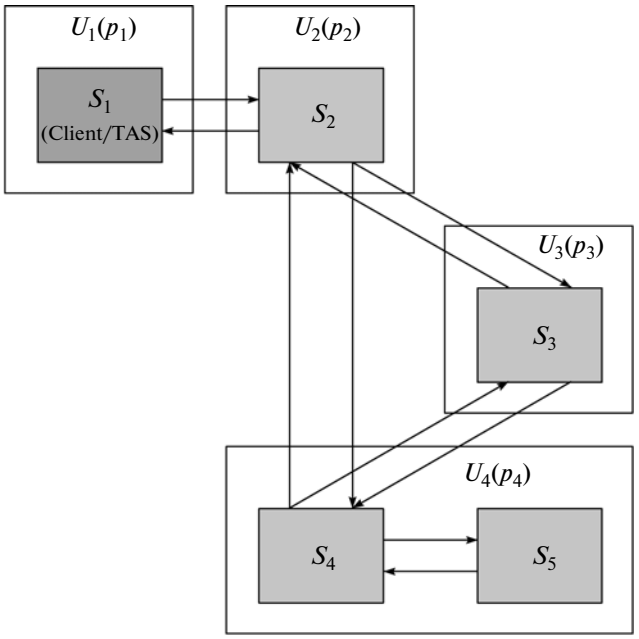

Fig. 3. Example of the model of the system as a graphical scheme.

model of measurements makes it possible to automate the comparison of the experimental results with the estimation criteria.

Below, we describe each phase in more detail.

3.1. Determination of Testing Goals

The testing goals are determined on the basis of the client's needs concerning the estimation or prediction of the operational characteristics of the IS and the corresponding nonfunctional requirements.

The client cannot always fully determine the testing goals. In the course of accommodation of the main goal with the client and during the development of the testing program and procedure, one can reveal secondary testing goals and limitations that need to be again approved by the client. The metamodel of requirements helps at the early stage form a questionnaire for the client and fully determine the testing goals, which reduces the number of iterations in goal approval procedure and the total time taken by the formation of the primary and secondary performance testing goals.

3.1.1. Construction of the model of requirements. At the stage of determination of testing goals, the metamodel of requirements makes it possible to analyze the object under test and construct a primary model of requirements, which is then discussed with the client, corrected, and approved.

The model of requirements is a verbal description of nonfunctional requirements grouped with respect to the types and objects to which the requirements are applied. This model contains also criteria for estimating the characteristics and calculated indicators resulting from performance testing.

The generation of the model of requirements is automated using the IBM Rational RequisitePro tool.

3.2. Development of Testing Program and Procedure

This is one of the most important and difficult phases of performance testing. At this phase, the models of the system, load, and measurements are generated on the basis of corresponding metamodels.

Three models are generated in parallel, and they are closely related with one another. For example, the interfaces of load feeding and the points of data collection can be determined only after the model of the system has been described.

When this phase is completed, all the approved models are included in the document called "Testing program and procedure", which is a plan of performance testing; this document must be approved by the client.

3.2.1. Construction of the model of the system. When the testing program and procedure are elaborated, a verbal and graphical model of the system is constructed, which then is discussed with the client, corrected, and approved.

The model of the system is a description of the object under test; it includes the software and hardware, their relationships, and the allocated resources (memory, processor, etc.).

Figure 3 shows an example of the model of the system as a graphical scheme of interaction between the software and computer systems.

The matrices K_S and K_U for this scheme are

$$K_S = \begin{pmatrix} 0 & 1 & 0 & 0 & 0 \\ 1 & 0 & 1 & 1 & 0 \\ 0 & 1 & 0 & 1 & 0 \\ 0 & 1 & 1 & 0 & 1 \\ 0 & 0 & 0 & 1 & 0 \end{pmatrix} \text{ and } K_U = \begin{pmatrix} p_1 & 0 & 0 & 0 \\ 0 & p_2 & 0 & 0 \\ 0 & 0 & p_3 & 0 \\ 0 & 0 & 0 & p_4 \\ 0 & 0 & 0 & p_4 \end{pmatrix},$$

where p_i is the vector of performance characteristics describing the amount of resources allocated for the given software in the computer system.

3.2.2. Construction of the model of load. At the phase of the testing program and procedure development according to the declared goals, a model of load is developed, which is then discussed with the client, corrected, and approved. The model must incorporate (usually the limiting) values of load that are predicted by the client and the configuration of the load sources that will be imitated in the course of testing. In fact, a load scenario meeting the goals of performance testing is developed in cooperation with the client. As a result of detailed examination of the scenario using the

metamodel of load, a model of load is constructed where all the load parameters are determined.

The model of load is a description of the test data and rules of their arrival to the system. In fact, the initial data for the load generation tool are formed based on the model of load.

3.2.3. Construction of the model of measurements. At the phase of the development of the program and procedure, a verbal model of measurements is formed according to the declared goals, which is then discussed with the client, corrected, and approved.

The model of measurements is a description of the characteristics to be collected and the methods used for their collection and interpretation.

The model of measurements determines the method and tools for collecting the values of measured variables. In this case, it is very convenient to use the metamodel of measurements, which incorporates the standard measurement procedures that are normally supported by the tools that are available on the market. The methods for collection of measured values can be very different depending on the testing goals.

At large, these models can be classed into methods of nondestructive control, methods of destructive control, and mixed methods.

In the case of nondestructive control, the measured values are collected using the tools available in the operational environment of IS and automated testing tools. In the case of destructive control, the measurement tools are partially embedded into the application software of the IS. In the case of mixed control, the measurement tools can be embedded into both system tools and application software of IS.

In an ideal case, the tools for collecting the measured values should not affect the system under test. When it is necessary to use the measurement tools that affect the system under test (to set interrupts or triggers), special experiments are performed to assess the degree of influence of the measurement tools on the performance of the object under test. The resulting estimates for the degree of influence are later used to interpret the results.

3.3. Preparation for Testing

Performance testing requires the preliminary execution of the following tasks:

– preparation of the workbench;

– preparation of the test data;

– preparation of the load feeding tools;

– preparation of the measurement tools.

The preparation of the test workbench is performed according to the approved model of the system and incorporates the organization of the software hardware and the fulfillment of the additional limitations (for example, time synchronization on all the computers used in the object under test).

For the testing preparation phase, the workbench check procedure based on the model of the system is automated. Errors in the preparation of the workbench can lead to the failure of the load experiment because the fact that the workbench is partially nonoperational can be detected only after the experiment has already started. In addition, the erroneous setting of the workbench can distort the results of performance testing, which can lead to the need of repeating the experiment. The availability of a formalized model of the system made it possible to automate the workbench checking, reducing the check time. If errors in the workbench configuration, in setting of its hardware and software are found, the automated checking can be repeated many times, which is not costly because this procedure is automated.

The test data for performance testing are prepared in line with the adopted model of load. The test data either are generated or real data with suitable characteristics are used.

At the testing preparation phase, the planning of the load experiment and generation of test data on the basis of the model of load are automated. In test planning, the temporal and quantitative parameters of the test data flows and the load feeding scenarios are determined. The automation tool for the load experiment planning makes it possible to input the corresponding initial data concerning the coming test into prescribed screen forms. Based on this information, the automated generation tool prepares test data. The automated generation of test data ensures that they are fully consistent with the current state of the system under test. Thus, it is guaranteed that the test data will not be rejected by the system and will be processed just like real data.

Load feeding (and the corresponding tools) are adjusted using the model of the system.

When the data collection tools are prepared, the model of measurements is used to configure the information suppliers and receivers.

The preliminary phases yield a workbench (at its initial state) and test data that are ready for feeding .

3.4. Load Feed

At the load feeding phase, the system under test receives the test data prepared according to their distribution laws described in the model of load.

The load feeding phase is automated using the testing tools that can form different types of load according to the law of their distribution in time. The load as messages is delivered from an earlier prepared test database. The load can be applied in the form of HTTP and SQL traffic. The configuration of traffic is generated on the basis of the preliminary automated analysis of the query codes sent by workstations of the system under test when operations are executed by users. The load in the form of events is generated by

Table 2. Duration of work stages

Phase	Duration	
	before implementing the technology	at present
Determination of testing goals	2 weeks	1 week
Development of testing program and procedure	5 weeks	2–3 days
Preparation for testing	3 weeks	2 days
Load feed Data collection	depending on the experimental design	
Interpretation and analysis of results	1 week	3–4 hours

emulating the operator actions using the IBM Rational Robot testing tools.

3.5. Data Collection

The data collection is performed using measurement tools. At the data collection phase, primary information about the values of measured indicators (such as processor workload, memory usage, transaction arrival times, etc.) is automatically collected. All the collected data are stored in the operational database for subsequent processing.

Depending on the collection methods specified in the model of measurements, the data can be collected both during testing as they arrive and after testing. The first variant is preferable because the collected data as they arrive can be used for controlling the test.

The data collection continues until the testing process satisfies a measurement termination criteria (determined in the model of measurements).

The primary data processing also depends on the processing procedure described in the model of measurements and can be executed both as data arrive and after the experiment as applied to the whole set of collected data.

3.6. Interpretation and Analysis of Results

The testing results are interpreted according to the model of measurements for further analysis: the indicators are calculated and the values of operational characteristics of the IS are estimated.

Based on the model of measurements, a multistep automated processing is performed on the primary data. First, the calculated indicators are generated on the basis of the rules and algorithms specified in the model of measurements. These indicators are placed into a data storage for subsequent analysis. Based on these data, analytical reports are generated describing the values of the calculated indicators and criteria for their estimation in line with the model of measurements and model of requirements. Because the data are stored, one can not only analyze the results of the load experiment but also perform a comparative analysis of several regression tests.

When the interpretation of the results is completed, conclusions are drawn on the correspondence of the IS with the testing goals taking onto account the limitations and criteria of the model of requirements. The results of performance testing are documented in a log sheet.

4. CONCLUSIONS

In performance testing, it is very important that both the client and the contractor develop a common understanding on the problem statement. This often becomes a long and cumbersome process because a common conceptual base must be worked out.

Even if there is need in performance testing, the goals of different experiments may vary because the client has different difficulties in connection with operational characteristics. Some of the typical problems are as follows:

– Can the existing hardware ensure the prescribed performance of the IS in the nearest future (1–2 years) provided that the annual growth of load is known?

– Can the IS that ensures the prescribed capacity when the load grows simultaneously ensure reactivity within the prescribed limits?

– Are there signs of degradation after the software was modified?

There are also other problems.

A large number of load experiments confirmed that many typical performance testing tasks can be unified.

The technology described in this paper has been used for 4 years to test several banking systems operating on different platforms—Windows, HPUX, zLinux, and z/OS. In different load testing experiments, up to 40 applications and 40 databases were run simultaneously. The maximum database size dealt by the applications under test was as large as 2.5 Tb.

Earlier, the experiments of this scale in client companies required large manual work. The load generation and application required a record of existing load flows.

No tests for the estimation of the expected characteristics of an IS could be peformed within tight time limits and at a relatively small cost (5–10 man–months).

The assessment of situations using conventional methods actually was rarely performed. No systematic predictive analysis of the degradation of performance characteristics after software modification was performed because of its high cost.

The technology proposed in this paper made it possible to perform during the last 4 years the following experimental studies (summary data):

– more than 40 load experiments;

– more than 75 million of generated messages;

– about 2 million messages (on the average) per experiment.

Table 2 presents data on the average duration of works before the implementation of the proposed technology and with the application of this technology.

The data presented in Table 2 show that the proposed technology and its automation tools make it possible to prepare and perform performance testing within tight time schedules that are suitable for clients. The number of repeated experiments to obtain the necessary characteristics was reduced from 2–3 to 1.

When large systems operating on costly hardware are tested, it is important to minimize such risks as the shift of testing schedule, the need for repeated tests, etc. These risks normally emerge because the client and the contractor have different views on the testing goals and due to errors in planning the experiment.

The proposed technology allowed us to reduce these risks, the duration and labor intensity of performance testing, and, consequently, its cost.

This study presents a technology for the organization and execution of automated performance testing for the class of ISs. The technology is based on the use of metamodels describing the requirements for performance testing, properties of the object under test, and the load. Within the frameworks of this technology, one can use metamodels to construct stage-by-stage models that are clear for the client and that can be used accommodate with the client on the main experimental parameters and result estimation criteria. Using these models, one can configure the tools; then, the experiment is performed automatically under the control of the performance testing tools. The technology covers all aspects of planning, load experiment, and analysis of its results. The use of models substantially (several times) reduces the labor intensity of the performance testing compared with our previous experience and makes it possible to reuse the prepared experiment (for example, in the control of IS degradation in its lifecycle).

REFERENCES

1. Kim, G.-B., A Method of Generating Massive Virtual Clients and Model-Based Performance Test, *Proc. of the Fifth Int. Conf. on Quality Software*, 2005, pp. 250–254.
2. Xing, C., Zhang, G., and Chen, M., Research on Universal Network Performance Testing Model, *Int. Symp. on Communication and Information Technologies*, 2007, pp. 780–784.
3. Kostogryzov, A., Panov, V., Pozin, B., and Sablin, V., Mathematical Modeling of Processes in Systems Life Cycles in Compliance with Standards Requirements of ISO/IEC 15288 and ISO/IEC 12207, *Proc. of the joint event "Software and System Convergence" SPINCOSE*, Montreal, Canada, 2003.
4. Kozlov, A.N. and Pozin, B.A., Technology of Complex Performance Testing of Banking System, *Proc. of the Int. Conf. "Reengineering of Business Processes Based on Modern Information Technologies for Knowledge Management System"*, Moscow, 2006, pp. 130–131.
5. Information Technology, Software Engineering, Software Measurement Process, *ISO/IEC 1539.*
6. Oliveira, F., *Performance Testing: an Introduction*, FACIN-PUCRS, 2008.