ТЕОРЕТИЧЕСКИЙ И ПРИКЛАДНОЙ НАУЧНО-ТЕХНИЧЕСКИЙ ЖУРНАЛ

Издается с сентября 2010 г.

Журнал издается при поддержке Отделения математических наук РАН, Отделения нанотехнологий и информационных технологий РАН, МГУ имени М.В. Ломоносова, МГТУ имени Н.Э. Баумана, ОАО "Концерн "Сириус".

СОДЕРЖАНИЕ

Кознов Д. В., Кудрявцев Д. В., Григорьев Л. Ю., Гагарский Р. К., Романовский К. Ю. Архитектура средств графического бизнес-моделирования в технологии ОРГ-Мастер 3

Болотова С. Ю. Реализация многопоточности в релевантном LP-выводе 12

Сирота А. А., Титов К. А. Архитектура распределенной информационной системы для поддержки технологий создания цифровых водяных знаков 19

Афонин С. А., Гаспорянц А. Э. Разрешение неоднозначности авторства публикаций при автоматической обработке библиографических данных. 25

Ефанов Д. В., Рощин П. Г. Вопросы реализации мандатной модели многоуровневого разграничения доступа в графической системе 29

Язов Ю. К., Сердечный А. Л. К разработке методов количественного оценивания эффективности ложных информационных систем 33

Иванова К. Ф. Оценка объединенного множества решений задачи на основе интервальной модели Леонтьева 40

Журнал зарегистрирован в Федеральной службе по надзору в сфере связи, информационных технологий и массовых коммуникаций. Свидетельство о регистрации ПИ № ФС77-38590 от 24 декабря 2009 г.

Журнал распространяется по подписке, которую можно оформить в любом почтовом отделении (индексы: по каталогу агентства "Роспечать" — **22765**, по Объединенному каталогу "Пресса России" — **39795**) или непосредственно в редакции.
Тел.: (499) 269-53-97. Факс: (499) 269-55-10.
Http://novtex.ru/pi.html E-mail: prin@novtex.ru

Журнал включен в систему Российского индекса научного цитирования.
Журнал входит в Перечень научных журналов, в которых по рекомендации ВАК РФ должны быть опубликованы научные результаты диссертаций на соискание ученой степени доктора и кандидата наук.

THEORETICAL AND APPLIED SCIENTIFIC AND TECHNICAL JOURNAL

SOFTWARE ENGINEERING

PROGRAMMNAYA INGENERIA

№ 1
January
2014

Published since September 2010

CONTENTS

Koznov D. V., Kudryavtsev D. V., Grigoriev L. Yu., Gagarskii R. K., Romanovskii K. Yu. An Architecture of Visual Modeling Tool of ORG-Master Toolset 3

Bolotova S. Yu. Multi-threaded Relevant LP-inference Implementation 12

Sirota A. A., Titov K. A. The Architecture of Distributed Information System to Support the Digital Watermarking Technology 19

Afonin S. A., Gaspariants A. E. Scientific Article Authorship Disambiguation for Automated Bibliographic Records Processing 25

Efanov D. V., Roschin P. G. Implementation Issues of Mandatory Multilevel Access Control Model in Graphical System 29

Yazov Y. K., Serdechnyy A. L. By the Development of Quantitative Methods of Effectiveness Honeynets Evaluation 33

Ivanova K. F. Estimation of the United Solution Set of an Interval Leontjev's Model 40

Information about the jornal is available online:
http://novtex.ru/pi.html, e-mail:prin@novtex.ru

УДК 004.4'22

Д. В. Кознов[1], канд. физ.-мат. наук, доц., e-mail: dkoznov@yandex.ru,
Д. В. Кудрявцев[1, 2], канд. техн. наук, доц., вед. консультант,
Л. Ю. Григорьев[2], ген. директор, **Р. К. Гагарский**[3], ген. директор,
К. Ю. Романовский[1], канд. физ.-мат. наук, ст. препод.,
[1] Санкт-Петербургский государственный университет,
[2] "Бизнес Инжиниринг Групп", г. Санкт-Петербург,
[3] ООО "Когнитивные Финансовые Технологии", г. Санкт-Петербург

Архитектура средств графического бизнес-моделирования в технологии ОРГ-Мастер[1]

Моделирование архитектуры предприятий (Enterprise Architecture Management, EAM) является областью исследований и одновременно практической деятельностью, которая направлена на структуризацию и преобразование структуры и функций бизнес-компаний на основе единого целостного подхода. ОРГ-Мастер является российским EAM-средством, имеющим 15-летнюю успешную историю развития и использования. В данной работе представлена архитектура средств визуального моделирования пакета ОРГ-Мастер, состоящая из следующих компонентов: внутренние редакторы — для визуального моделирования в рамках пакета ОРГ-Мастера; внешние редакторы — средства для разработки "легковесных" графических редакторов на базе Microsoft Visio для работы с заказчиками в рамках конкретных консалтинговых проектов; набор полноценных графических редакторов, используемых в различных проектах без изменений; графические отчеты — технология для разработки диаграммных отчетов, автоматически генерируемых по модели ОРГ-Мастера и предназначенных для презентативных задач; импорт из других визуальных средств.

Ключевые слова: *архитектура предприятия, моделирование архитектуры предприятия, онтологический инжиниринг, онтологии, бизнес-процессы, предметно-ориентированное моделирование, визуальное моделирование, модельно-ориентированная разработка, DSM-подход, интеллект-карты, карты памяти, и-карты*

D. V. Koznov, D. V. Kudryavtsev, L. Yu. Grigoriev, R. K. Gagarskii, K. Yu. Romanovskii

An Architecture of Visual Modeling Tool of ORG-Master Toolset

Enterprise Architecture Management (EAM) is a growing area, which supports enterprise transformations and provides business-IT alignment. ORG-Master is a Russian EAM tool, which is successfully applied in Russian industry during last 15 years. The paper presents of ORG-Master visual modeling tool architecture, which consists of the following parts: internal editors (for visual modeling inside of ORG-Master); external editors (to support of lightweight customer-oriented graphical notations); graphical reports (read only visual models, which can be automatic generated under ORG-Master model); import.

Keywords: *enterprise architecture, enterprise architecture modeling, enterprise architecture management, EAM, EAM tools, DSM, domain-specific modeling, mind map, concept map, BPMN, business processes*

[1] Работа выполнена при поддержке гранта РФФИ 12-01-00415-а.

Введение

В настоящее время бизнес-организациям необходимо постоянно развиваться, чтобы своевременно реагировать на изменения рынка, среды и технологий. Для проведения соответствующих изменений принято выделять так называемую архитектуру предприятия (Enterprise Architecture, EA), отражающую его устройство. Эта архитектура предназначена для анализа, проектирования изменений и, собственно, реализации самих изменений предприятия [1—3]. Для моделирования EA (Enterprise Architecture Modeling, EAM) и дальнейшей работы с этими моделями применяют специальные программные средства — инструменты управления архитектурой предприятия (EAM tools) [4, 5]. Они активно используют визуальное моделирование, т. е. "чертежи, позволяющие точно изображать хорошо структурированную информацию в понятной форме" [6], и, следуя традиции программной инженерии, для их разработки создаются различные графовые нотации. Данная область в настоящий момент активно развивается, о чем свидетельствует, например, отчет Gartner Group за 2012 г. [7].

В России в течение последних 15 лет активно развивается и используется программно-методический комплекс ОРГ-Мастер, предназначенный для предоставления сервисов российским предприятиям в области моделирования EA [8—17]. ОРГ-Мастер ориентирован не только на создание процессных моделей, но также и на разработку целостных моделей всего бизнеса компании на основании ее ключевых компетенций. Особенностью ОРГ-Мастер является использование таблично-диалогового способа представления информации, а также простого и гибкого языка моделирования. Эти средства при необходимости сочетают со средствами визуального моделирования, что позволяет не только рисовать "красивые картинки" для начальства, но также оперативно разрабатывать детальные модели бизнеса, отделяя уровень внутреннего (т. е. рабочего) представления информации от внешнего, презентационного, предназначенного для клиентов [12].

В связи с тем, что визуальное моделирование в ОРГ-Мастере не используется как ведущее средство работы с информацией, возникает необходимость в специальной концепции, которая бы определяла место диаграмм в процессе разработки проектов по созданию архитектур бизнес-предприятий (далее — EA-проектов). Также необходимо максимально ясно описать различные сценарии использования визуальных средств: диаграммы должны встраиваться в процесс разработки архитектуры предприятия, а средства для их поддержки — максимально "бесшовно" интегрироваться с базовым EAM-пакетом.

В данной работе предложена архитектура средств визуального моделирования, удовлетворяющая этим требованиям. Ее основными составляющими являются следующие компоненты: внутренние редакторы — для моделирования в рамках пакета ОРГ-Мастер; внешние редакторы — технология разработки "легковесных" графических редакторов на базе Microsoft Visio для конкретных EA-проектов, а также набор полноценных графических редакторов, используемых в различных проектах без изменений; графические отчеты — технология разработки диаграммных отчетов, автоматически генерируемых по модели; инструменты импорта из других средств моделирования. Представлена пилотная реализация данной архитектуры на базе известной графической среды Microsoft Visio.

Программно-методический комплекс ОРГ-Мастер

Опишем технологические аспекты ОРГ-Мастера — язык моделирования, программные средства и основной сценарий использования. Детали методологии читатели могут найти в работах [8—13].

Язык моделирования ОРГ-Мастер подробно изложен в работах [13—17][2] и является компактным средством для создания иерархических моделей с большим числом связей и не очень большим числом атрибутов. Именно это требуется при создании EA-моделей, так как вся информация, характеризующая бизнес-компанию, является сильно связной, и в то же время архитектурная модель не подразумевает большого числа атрибутов у отдельных сущностей. Архитектурная модель не является ни репозиторием предприятия, ни схемой базы данных некоторой информационной системы. Последнее связано с тем, что параллельно с такой моделью на предприятии обычно существуют различные базы данных — сотрудников, клиентов, заказов и т. д. Например, архитектурная модель может содержать организационную структуру предприятия — все подразделения и их связи, а также возможные должности сотрудников. Но в ней не будут указаны такие атрибуты, как ФИО сотрудника, телефон, домашний адрес и пр., также не будет в ней и записей о конкретных сотрудниках. Отметим, что в язык моделирования ОРГ-Мастера не входят понятия предметной области EA, такие как функциональная система, бизнес-компетенция и пр. (последние входят в различные расширения языка).

[2] В работе [13] этот язык называется "Формализованный язык описания модели организации (ORLAN)", в работе [14] — "Язык классификаторов и проекций для наполнения баз знаний", в работе [15] — "ORG-Master metamodeling language". В дальнейшем будем называть этот язык так же как в данной работе.

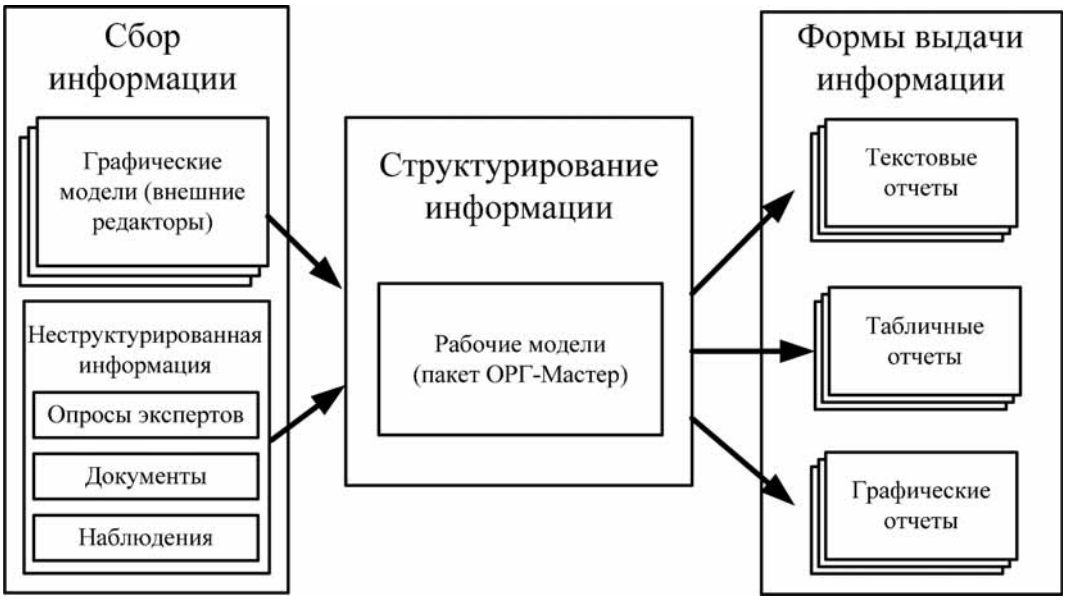


Рис. 1. Базовый сценарий использования ОРГ-Мастера

Программные средства ОРГ-Мастера состоят из базового средства моделирования и набора дополнительных утилит. Базовое средство включает в себя репозиторий моделей и среду для удобной работы с этим репозиторием, а также генератор отчетов в форматах Microsoft Word, Excel, HTML, средства для импорта данных из различных источников и процедуры слияния двух версий моделей, а также другие сервисные функции.

Рассмотрим подробно главный сценарий использования ОРГ-Мастера. Для того чтобы собрать информацию о предприятии в рамках EA-проекта, аналитикам приходится много общаться с работниками предприятия, изучать различные рабочие документы, а также непосредственно наблюдать за работой предприятия. Бывает также, что первичное осмысление и формализация этой информации уже проведены аналитиками компании — тогда данная информация, как правило, представлена в виде набора диаграмм (в российских компаниях для этого используется в большинстве случаев средство Microsoft Visio). Далее вся эта информация вводится в пакет ОРГ-Мастер и должным образом структурируется. После этого создается итоговое представление информации в терминах различного вида отчетов. Последние могут формироваться в итоговые отчетные документы, но могут также выкладываться на специальный портал. На рис. 1 представлена схема этого процесса. Отметим, что процесс является итеративным, поэтому ОРГ-Мастер, во-первых, поддерживает автоматические переходы с этапа на этап, а во-вторых, обеспечивает циклическую разработку, т. е. синхронизацию различных активов с сохранением информации, которая внесена в одно представление и отсутствует в другом.

Пакет Microsoft Visio как платформа разработки графических средств

Существует большое число технологий для разработки графических средств (так называемые DSM-платформы [18]). Можно упомянуть инициативу сообщества Eclipse Modeling Project, объединяющую около 100 открытых проектов в области моделирования, в том числе такие зрелые технологии, как GMF, EMF, GEF [19]. Широко известна также среда разработки MetaEdit+ [20], главными идеологами которой являются Steven Kelly и Juha-Pekka Tolvanen (университет Ювяскюля, Финляндия), которые на настоящий момент являются, пожалуй, главными идеологами и популяризаторами DSM-подхода[3]. Отметим также пакет QReal [22—24], развиваемый в Санкт-Петербургском государственном университете и являющийся идейным продолжением более ранних проектов в области визуального моделирования Real/RTST++ [25] и RTST [26], имеющих большую историю успешных применений, уходящую еще во

[3] Отметим прежде всего монографию [21], которая на сегодняшний день является самым полным источником о различных аспектах реализации и промышленного использования предметно-ориентированных решений.

времена СССР и закрытых военных разработок.[4] Обзоры DSM-платформ на русском языке можно найти в работах [18, 27].

В рассматриваемом проекте в качестве DSM-платформы были выбраны пакеты Microsoft Visio и Microsoft Visual Studio. Этот выбор был обусловлен следующими причинами.

- Большинство пользователей графических средств технологии ОРГ-Мастер — менеджерский состав средних и крупных российских бизнес-компаний — знают Visio или хотя бы слышали о нем. Этот пакет легко доступен (входит в состав Microsoft Office) и имеет большое число нотаций и решений для самых разных областей деятельности. Кроме того, существует много свободно распространяемых надстроек для Visio. Наконец, можно легко создать в Visio свой собственный графический редактор. В силу всего вышесказанного решения, созданные на основе этого пакета, не вызывают у пользователей боязни и первичного отторжения, что очень важно при реализации EA-проектов.
- Visio является удобным, многофункциональным и зрелым пакетом, созданные на его основе графические редакторы наследуют много важных и необходимых для работы функций, таких как готовый пользовательский интерфейс (палитра, главное меню и пр.), рабочая область с настраиваемыми свойствами, автоматическим увеличением в случае выхода за границу листа, многофункциональная и настраиваемая печать и т. д.
- Пакет содержит удобные средства для разработки расширений — как для быстрого создания простых редакторов (с помощью настроек и встроенного скриптового языка VBA), так и для более серьезных решений с помощью открытого программного интерфейса и специальных средств связи с Microsoft Visual Studio (в частности, приложения, создаваемые с помощью Visual Studio, имеют возможность "видеть" важнейшие события Visio и реагировать на них).

Предыдущий опыт авторов по использованию Visio [28, 29] послужил дополнительным аргументом в выборе Microsoft Visio в качестве DSM-платформы.

Описание архитектуры

ОРГ-Мастер не использует диаграммы в качестве базовых средств моделирования. В качестве последних используется таблично-диалоговое моделирование, и, таким образом, визуальные средства должны быть подходящим образом интегрированы с ними[5]. При этом они оказываются существенно различными — по назначению, по функциональным возможностям, а также по способам интеграции с пакетом ОРГ-Мастер. Представим средства, каждое из которых подробно рассматривается в следующих разделах.

- Внутренние редакторы — являются частью поставки базовых средств моделирования базового EAM-пакета (в данном случае — ОРГ-Мастера), используются для выборочного создания фрагментов моделей (установление связей, просмотр модели и ее фрагментов и т. д.).
- Внешние редакторы — предназначены для клиентов, обеспечивают удобство ввода информации в ОРГ-Мастер, могут дорабатываться для нужд конкретного EA-проекта; возможно создание новых внешних редакторов.
- Графические отчеты — конечные диаграммные отчеты для заказчика (для вставки в отчетные документы или выгрузки на web-портал).
- Средства импорта из других визуальных средств (Visio, CMap [32] и пр.).

Внутренние редакторы

Данные редакторы встроены в ОРГ-Мастер и предназначены для удобства редактирования и просмотра модели аналитиками. Важно, что они предоставляют средства для работы с моделью ОРГ-Мастера в терминах базового языка — классификаторов, проекций и т. д., и не касаются бизнес-расширений

[4] Отметим, что многие программные средства и, в частности, EAM-пакеты, такие как Mega [30] и ARIS [31], обладают развитыми средствами расширения — т. е. их можно надстраивать и дорабатывать. Однако не будем вносить такие средства в список DSM-платформ, так как, во-первых, их использование ограничено предметной областью, а во-вторых, они реализуют конкретный метод в рамках этой области. Нас же интересуют средства, которые позволят реализовать собственные решения в нужной предметной области. Так, все упомянутые DSM-платформы могут использоваться как в области разработки ПО, так и в EA-проектах, а также во многих других областях (за исключением, быть может, MetaEdit+ [20], который нацелен на разработку визуальных решений именно в области разработки ПО и поддерживает средства для разработки генераторов кода по моделям и другие специфические черты).

[5] Решение REAL-IT, предоставляющее средства автоматизированной разработки приложений интенсивной работы с данными и поддерживающее широкий спектр генерационных возможностей, также основывается не на визуальном моделировании, а на таблично-диалоговом задании исходных данных для генерации [33—35]. И так же как ОРГ-Мастер, REAL-IT частично использует визуальные модели, например, для создания схем баз данных. Так что можно говорить о специфической парадигме моделирования, когда визуальные модели сосуществуют с моделями не визуальными, т. е. визуализируется только та информация о системе, с которой удобно работать в таком виде. С остальной информацией работа происходит с помощью тщательно спроектированного диалогового интерфейса. В этом случае репозиторий с моделью выступает в виде обычной базы данных, для которой разработан специальный оконный пользовательский интерфейс.

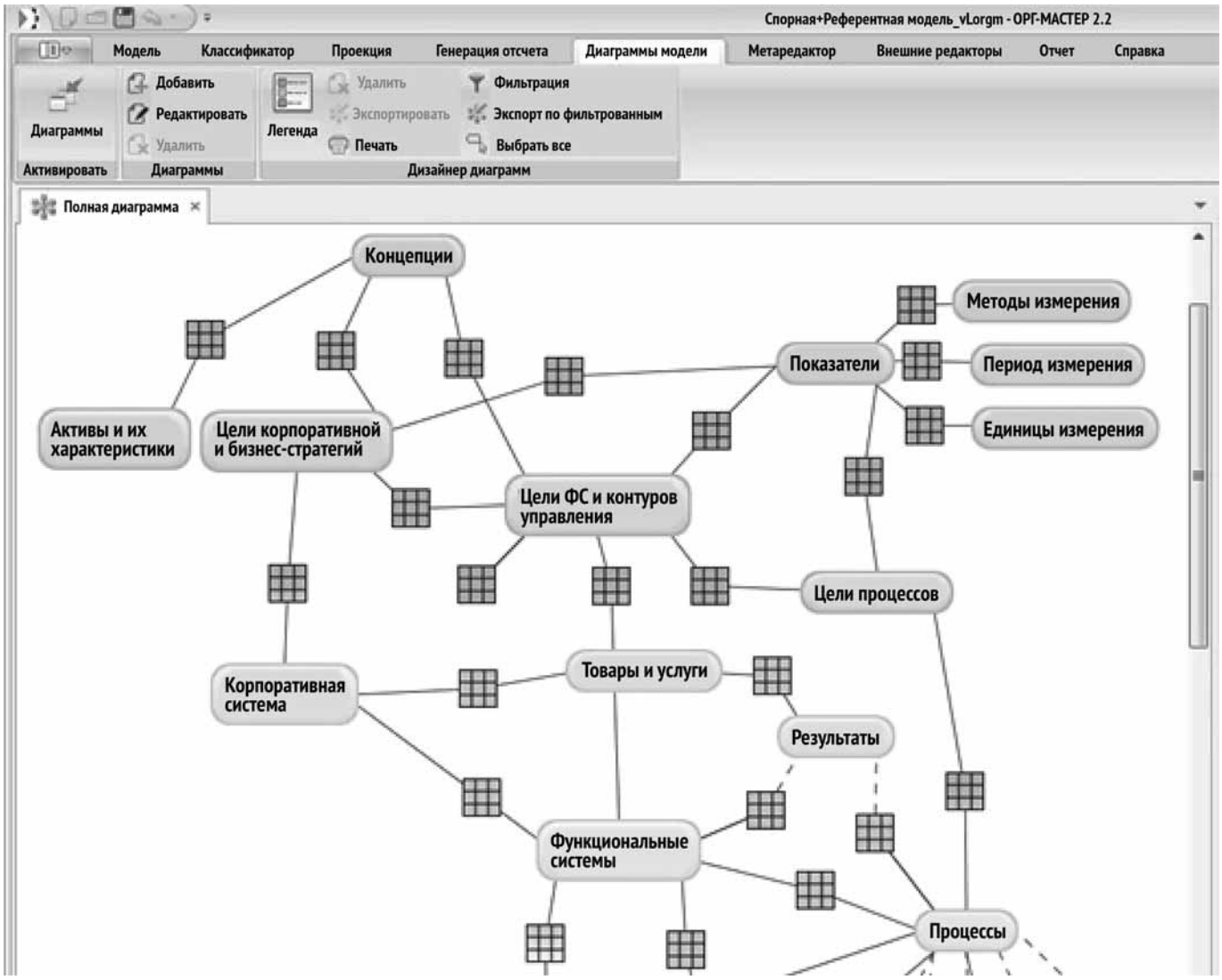


Рис. 2. Пример внутреннего редактора — редактор модели

(документов, процессов, ценностей и т. д.). Эти редакторы предназначаются для локального использования, т. е. их место в пакете ОРГ-Мастер, а также в процессе моделирования, строго определено: их используют для визуализации той информации, которую удобно так представлять в процессе разработки модели. Число и функциональные возможности таких редакторов ограничены ввиду неграфической концепции моделирования. Кроме того, их разработка является достаточно трудоемкой в силу невозможности (в виду лицензионных соглашений и интеграционных проблем) использовать Microsoft Visio и другие DSM-платформы.

В настоящий момент реализован один внутренний редактор — редактор модели, который позволяет создавать и просматривать структуру модели в ОРГ-Мастере в терминах классификаторов и проекций (пример представлен на рис. 2).

Внешние редакторы

Основная задача внешних редакторов заключается в гибкой поддержке различных графических нотаций, необходимость в которых возникает в связи с определенным EA-проектом или серией проектов. Ввиду отсутствия стандартизации в данной области, а также в силу ее большой вариативности, надобность в поддержке новых нотаций возникает часто.

Для решения этой задачи пойдем по пути создания гибкой технологии для быстрой реализации новых графических нотаций. Выделим два следующих источника возникновения новых нотаций:

- клиенты (работники бизнес-компаний, для которых выполняются EA-проекты), имеющие при-

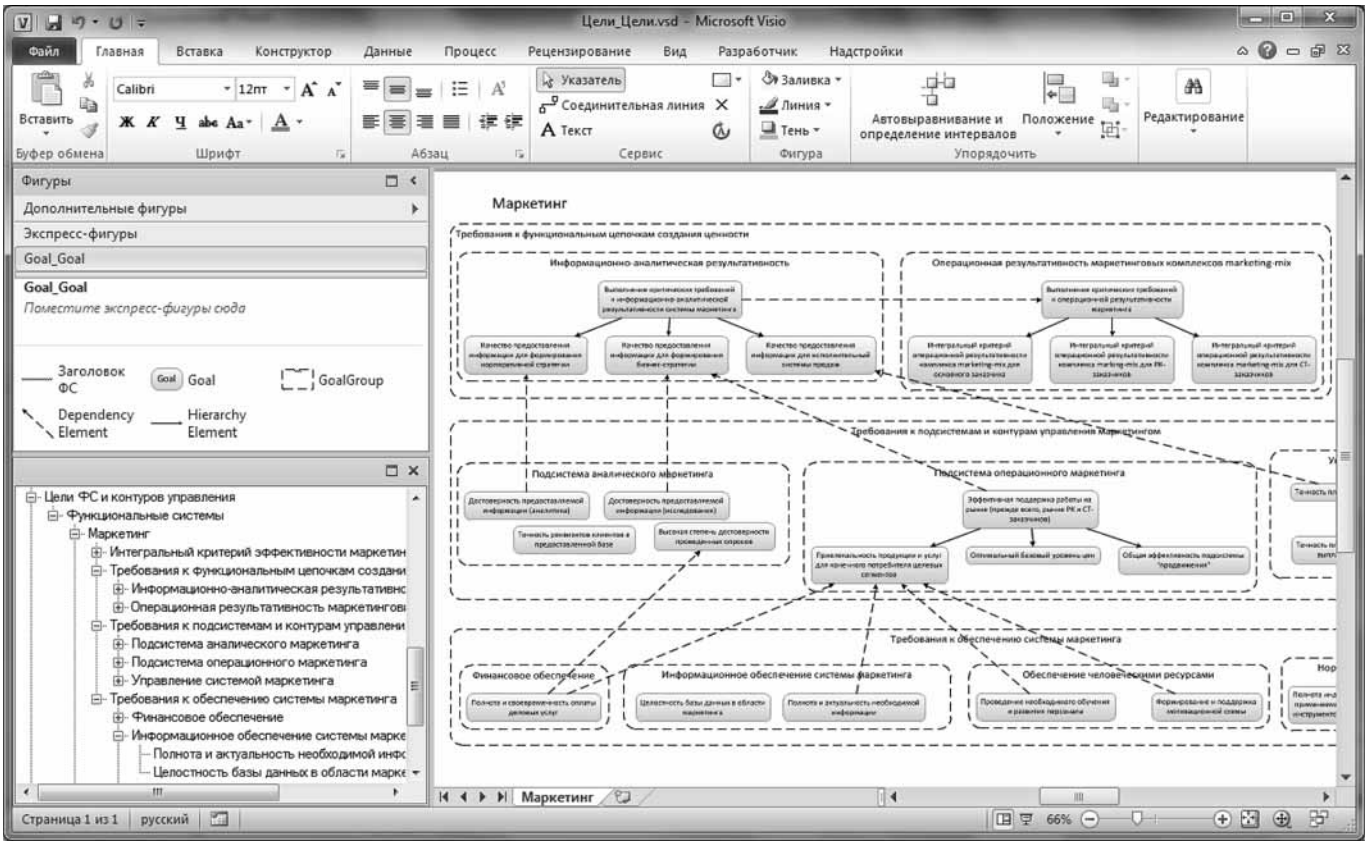


Рис. 3. Пример внешнего редактора — редактора стратегий

страстия к определенным графическим нотациям и отдельным Visio-палитрам; более того, такие клиенты часто предоставляют аналитикам большое число уже готовых диаграмм в Visio и хотят продолжать дальнейшую работу над формализацией бизнес-архитектуры своей компании в Visio в рамках этих нотаций;

- развитие методологии ОРГ-Мастера, что важно ввиду того, что область EA является молодой и бурно развивающейся; в связи с чем общепризнанных стандартов, в частности, в области моделирования, почти нет.

Внешние редакторы разделим на две группы: стандартные, которые используют в разных EA-проектах и доводят до состояния законченных продуктов; редакторы, предназначающиеся для отдельного EA-проекта, которые используют базовые функциональные возможности Visio и создаются на основе повторно используемых компонентов. Внешние редакторы не должны предоставлять пользователям дополнительных функций, кроме тех, которые дает Visio и повторно используемые активы, это важно, так как иначе разработка такого графического редактора рискует оказаться сопоставимой по затратам с тем проектом, для которого он предназначается: число специфических функций для произвольного графического редактора необозримо.

При разработке архитектуры внешних графических редакторов был сформулирован ряд следующих исходных требований.

- Внешний редактор и ОРГ-Мастер должны поддерживать двустороннюю циклическую разработку модели. Это требование стало следствием пожеланий заказчиков о том, что клиенты, получая внешний редактор, должны также получить и некоторый каркас модели ОРГ-Мастера, т. е. контекст моделирования. Таким образом, они должны моделировать не с "чистого листа". Далее, требовалось также поддержать генерацию из внешнего редактора в ОРГ-Мастер — клиентские модели должны стать достоянием опытных аналитиков, отвечающих за создание итоговой модели и дальнейших, следующих из нее, артефактов. Наконец, во многих случаях оказалось полезным поддерживать двустороннюю (клиенты/аналитик) работу с моделью через внешние редакторы, т. е. чтобы клиенты видели (в виде диаграмм) изменения в модели, которые сделали аналитики, и аналитики в свою очередь могли бы получать несколько последовательных версий моделей от клиентов (также в виде диаграмм). Это и есть полноценная циклическая

разработка, как это определено в работе [36]. Механизм циклической разработки, реализованный во внешних редакторах ОРГ-Мастера, был опробован авторами при создании редактора онтологий [29][6].

- Было необходимо реализовать браузер модели ОРГ-Мастера, который был бы доступен во всех внешних редакторах и позволял бы выбирать элементы из модели для размещения на диаграмме при помощи перетаскивания мышью.
- Последнее, естественное ограничение — разрабатываемое авторами решение было предназначено для интеграции только с ОРГ-Мастером, т. е. формат репозитория и средства программного доступа были фиксированы.

Отметим, что рассматриваемая архитектура не охватывает ряд важных вопросов, например, типовые окна для заполнения текстовых атрибутов элементов на диаграмме, средства декомпозиции больших моделей и т. д. Авторы планируют реализовать эти функциональные возможности для стандартных внешних редакторов.

На данный момент авторами реализованы три внешних редактора:

- карты стратегий, предназначенные для моделирования стратегий бизнес-компании (пример на рис. 3);
- диаграмма процессов (нотация предложена авторами);
- нотация "процесс—процесс—результат", предназначенная для верхнеуровневого взгляда на взаимодействующие процессы (нотация предложена авторами).

Графические отчеты

Графический отчет — это диаграмма, автоматически сгенерированная по модели ОРГ-Мастера в соответствии с определенной нотацией. Такие диаграммы предназначаются для включения в итоговый пакет документации по EA-проекту, а также могут использоваться при разработке портального решения с описанием такой архитектуры. Таким образом, модификации данных диаграмм ограничены — они либо вообще не должны меняться и используются as is, либо может изменяться только их внешний вид. Последнее важно, так как они предназначаются для клиентов и должны быть красивыми, а автоматическая раскладка может этого не обеспечить, и поэтому требуется "ручная" доводка. Кроме того, в случае, когда диаграммы достаточно сложны (например, диаграммы бизнес-процессов в нотации BPMN), то их автоматическая раскладка затруднена в принципе.

Однако как только разрешается менять "вручную" внешний вид автоматически сгенерированных диаграмм, сразу же становится необходима синхронизация этих диаграмм с моделью ОРГ-Мастера. Дело в том, что часто бывает трудно с первого раза создать окончательную версию графического отчета — после того как он создан, приходится тратить значительные усилия на улучшение его внешнего вида (раскладка, шрифты надписей и пр.), выявляются всякие мелочи в модели, которые нужно исправить и т. д. Когда соответствующий фрагмент модели представлен в графическом виде, становятся очевидными (как аналитиками, так и заказчику) недоработки и ошибки — от погрешностей русского языка в наименованиях модельных сущностей (здесь сущности, в отличие от идентификаторов сущностей ПО, часто имеют длинные названия, состоящие из нескольких слов), до пропущенных сущностей и других видов ошибок. Все эти изменения нужно вносить в исходную модель в ОРГ-Мастере, по которой были сгенерированы графические отчеты. Если после этого графические отчеты сгенерировать повторно, то будет потеряна вся сделанная прежде работа по украшению этих диаграмм.

В данный момент поддерживается генерация графических отчетов в двух форматах: SVG (для использования диаграмм в портальных решениях), а также в формате Visio. Для последних поддерживается односторонняя синхронизация с моделями ОРГ-Мастера, необходимость которой была только что аргументирована. Подчеркнем, что эта синхронизация только в одну строну, т. е. предполагается, что изменять модель посредством создания новых сущностей в графических отчетах нельзя[7]. Это предохраняет от того, чтобы вместо графических отчетов разрабатывать полноценные графические редакторы (в одном из EA-проектов, где использовались излагаемые в статье идеи, число нотаций, которые понадобились заказчику и которые были реализованы в виде графических отчетов, составило девять — создать в рамках одного EA-проекта столько же графических редакторов было бы слишком трудоемко).

На рис. 4 (см. третью сторону обложки) представлен пример графического отчета из проекта по разработке архитектуры деятельности правительства города Москвы, в котором технология ОРГ-Мастер и ряд решений, представленных в данной работе, активно ис-

[6] Про разные варианты понимания пользовательского функционала для поддержки циклической разработки в современных UML-пакетах см. работу [39].

[7] Аналогичное решение было реализовано в проекте по поддержке концептуального проектирования при разработке визуальных языков, в рамках которого были проинтегрированы следующие продукты: Visio, Microsoft DSL Tools и Word [37]. Во всех трех продуктах можно было по-разному представлять метамодель нового визуального языка, в том числе в виде табличного документа в Word, но менять модель из Word не разрешалось — эти изменения, во-первых, не попадали в два оставшихся продукта, во-вторых, они терялись и в самом Word при синхронизации с Visio и Microsoft DSL Tools.

пользовалась [38]. На рис. 4 (см. третью сторону обложки) представлена контекстная диаграмма городского процесса "Передача земельного участка из собственности города Москвы в федеральную собственность". Данная диаграмма показывает, какую информацию и из каких ресурсов получает процесс, какую информацию и куда процесс передает, к какой сфере ведения и государственной функции относится процесс, а также кто отвечает за выполнение процесса. Раскладка данной диаграммы выполнена автоматически, без дополнительной "ручной" доводки[8].

Импорт из других визуальных средств

Предложенные выше средства решают не все потребности в моделировании, которые встречаются в EA-проектах. Выделим еще два случая, когда требуются дополнительные средства — они, по мнению авторов, не могут быть сведены к описанным выше.

Во-первых, часто клиентов удается убедить перейти на средства моделирования, поддерживаемые ОРГ-Мастером, и не использовать далее свои собственные нотации, в рамках которых они начали работу по описанию своей компании до начала EA-проекта с ОРГ-Мастером. Но при этом есть некоторое число созданных клиентами диаграмм (и часто это число велико), информацию из которых хотелось бы перенести в формирующуюся модель в ОРГ-Мастере. В большинстве случаев клиенты рисовали свои диаграммы в Visio и использовали какую-нибудь палитру, поставляемую прямо с продуктом.

Во-вторых, при моделировании очень удобно использовать нотации типа Mind Maps/Concept Maps[9] [32, 40], для которых существует много удобных, хорошо себя зарекомендовавших и широко распространенных программных средств, например, обзор средств Mind Map можно найти в работах [41, 42]. При этом сценарии использования таких средств могут сильно варьироваться — от конспектирования аналитиком бесед с клиентом в виде Mind Map-диаграммы, до создания развернутых концептуальных карт, задающих структуру целевой модели в ОРГ-Мастере (такой вариант использования дополнительных нотаций описан в работе [29]). Все такие диаграммы хотелось бы уметь автоматически переносить в ОРГ-Мастер, поэтому очень полезно иметь функцию импорта из этих средств. В настоящий момент авторами реализован только настраиваемый импорт из Microsoft Visio.

Важно отметить, что импорт не предназначен для поддержки синхронизации (в отличие от графических отчетов, которые могут поддерживать одностороннюю синхронизацию, или внешних редакторов, которые поддерживают двухстороннюю синхронизацию) и выполняет задачу разового переноса информации из разных источников в ОРГ-Мастер. Если, тем не менее, синхронизация нужна, то в случае с Visio нужно создавать соответствующий внешний редактор. В других же случаях такое решение может оказаться затруднительным, прежде всего в части реализации экспорта изменений из ОРГ-Мастера в сторонние средства в виду трудностей с возможностями программных интерфейсов таких средств (как правило, это касается средств программной отрисовки, но часто и online-доступа к элементам модели); Visio в данном случае является исключением и поддерживает соответствующие функциональные возможности.

Заключение

Представленная архитектура пока не реализована полностью, однако для каждой ее части уже создан и частично опробован в реальных проектах соответствующий набор программных средств. Ниже представлены дальнейшие направления развития данной архитектуры.

- Разработка средств слияния моделей: адаптация существующих алгоритмов слияния деревьев и XML-файлов, реализация удобного пользовательского интерфейса (в качестве примера см. реализацию известного алгоритма слияния XML-файлов 3DM в рамках Интернет-сервиса Comapping [43, 44]).
- Перевод средств моделирования бизнес-процессов ОРГ-Мастера на стандарт в этой области BPMN [45][10].
- Активная практическая апробация всех аспектов концепции с их исправлением, доработкой и дальнейшим развитием. По мнению авторов, подобные концепции могут быть только плодом интенсивной практической работы в реальных проектах.

[8] Диаграмма взята с внутреннего портала департамента информационных технологий города Москвы, в разработке которого использовалось решение, представленное в данной статье. Всего в данном проекте было сгенерировано около 5000 графических отчетов, которые в совокупности с таблицами и обычным текстом составили контент данного портала.

[9] Существует несколько вариантов русского перевода термина Mind Map — карты памяти, интеллект-карты, и-карты; Concept Map обычно переводится как концептуальные карты.

[10] Например, в России одним из самых известных приверженцев стандартов как в IT-образовании, так и в IT в целом является проф. А. Н. Терехов (см., например, его работу [46]). По мнению авторов, он совершенно прав, так как стандарты аккумулируют положительный опыт, полученный в науке и производстве. Однако есть и другие точки зрения — дискуссию о стандартизации в области разработки ПО см. в работе [47]. Авторы считают, что стандарты в IT нужно использовать (обязательно!), но не бояться их исправлять и дополнять, применяя в конкретных ситуациях. Эта точка зрения высказывается одним из авторов в работе [48] применительно к преподаванию в сфере IT.

Список литературы

1. **Данилин А., Слюсаренко А.** Архитектура и стратегия. "Инь" и "Янь" информационных технологий предприятия. М.: Интернет-Университет Информационных Технологий. 2005. 504 с.

2. **Зиндер Е. З.** Архитектура предприятия в контексте бизнес-реинжиниринга // Intelligent Enterprise. 2008. Ч. 1. № 4. С. 46; Ч. 2. № 7. С. 183.

3. **Op't Land M., Proper E., Waage M., Cloo J., Steghuis C.** Enterprise Architecture: Creating Value by Informed Governance. Berlin: Springer—Verlag, 2009. 146 p.

4. **Калянов Г. Н.** Архитектура предприятия и инструменты ее моделирования //Автоматизация в промышленности. 2004. № 7. С. 9—12.

5. **Buckl S.** 7 EAM tools: State-of-the-Art [Электронный ресурс]. Режим доступа: http://www.hpi.uni-potsdam.de/hirschfeld/teaching/past/itua12/media/Itua12_07_EAMTools.pdf, свободный. Яз. англ. (дата обращения 02.09.2013).

6. **Ross D. T.** Structured Analysis (SA): A Language for Communicating Ideas // IEEE Transactions Software Engineering. 1977. V. 3, № 1. P. 16—34.

7. **Bittler R. S.** Magic Quadrant for Enterprise Architecture Tools. Gartner, 2012. G00234030. 28 p.

8. **Менеджмент** по нотам: технология построения эффективных компаний / Под ред. Л. Ю. Григорьева. М.: Альпина Паблишерз, 2010. 694 с.

9. **Григорьев Л.Ю., Кудрявцев Д. В.** Организационное проектирование на основе онтологий // Научно-технические ведомости СПбГПУ. Сер. "Информатика. Телекоммуникации, Управление". 2012. № 1 (140). С. 22—27.

10. **Kudryavtsev D. V., Grigoriev L. Y., Kislova V. V., Zablotsky A. V.** Using ORG-Master for knowledge based organizational change // Information Theories & Applications. 2006. V. 13, N 2. P. 131—139.

11. **Kudryavtsev D. V., Grigoriev L. Y.** Ontology-based business architecture engineering technology // Frontiers in Artificial Intelligence and Applications, Vol. 213: New Trends in Software Methodologies, Tools and Techniques. Amsterdam: IOS Press, 2011. P. 233—252.

12. **Grigoriev L., Kudryavtsev D.** Non-diagrammatic method and multi-representation tool for integrated architecture and business process engineering // Proceesings of 15th IEEE Conference on Business Informatics (CBI 2013). Vienna, Austria. 15—18 July, 2013. Danvers: IEEE Computer Society Conference Publishing Services (CPS), 2013. P. 258—263.

13. **Кудрявцев Д. В.** Разработка моделей и методов обработки знаний в области организационного проектирования на основе онтологий: дис. ... канд. техн. наук: 05.13.01; 05.13.11. Санкт-Петербургский государственный политехнический университет. СПб. 2009.

14. **Григорьев Л. Ю., Заблоцкий А. А., Кудрявцев Д. В.** Технология наполнения баз знаний онтологического типа // Научно-технические ведомости СПбГПУ. Сер. "Информатика. Телекоммуникации. Управление". 2012. № 3 (150). С. 27—36.

15. **Grigoriev L., Kudryavtsev D.** ORG-Master: Combining Classifications, Matrices and Diagrams in the Enterprise Architecture Modeling Tool // Proceedings of the 4th Conference on Knowledge Engineering and Semantic Web, October 7—9, 2013. Communications in Computer and Information Science (CCIS) Series, Springer, 2013. P. 250—258.

16. **Кудрявцев Д. В., Кознов Д. В., Григорьев Л. Ю.** Новые возможности языка моделирования ОРГ-Мастер // Научно-технический вестник информационных технологий, механики и оптики. 2013. № 6 (86). С. 79 —85.

17. **Григорьев Л. Ю., Кудрявцев Д. В., Кознов Д. В.** Обзор языка моделирования ОРГ-Мастер // Системное программирование. 2013. Т. 8, №. 1. Принято к печати.

18. **Павлинов А. А., Кознов Д. В., Перегудов А. Ф., Бугайченко Д. Ю., Казакова А. С., Чернятчик Р. И., Иванов А. Н.** О средствах разработки проблемно-ориентированных визуальных языков // Системное программирование. 2006. Т. 2, № 1. С. 116—141.

19. **Сорокин А. В., Кознов Д. В.** Обзор Eclipse Modeling Project // Системное программирование. 2010. Т. 5, № 1. С. 6—32.

20. **MetaEdit+.** URL: http://www.metacase.com/

21. **Kelly S., Tolvanen J.-P.** Domain-Specific Modeling: Enabling Full Code Generation. John Wiley & Sons, 2008. 340 p.

22. **Терехов А. Н., Брыксин Т. А., Литвинов Ю. В., Смирнов К. К., Никандров Г. А., Иванов В. Ю., Такун Е. И.** Архитектура среды визуального моделирования QReal // Системное программирование. 2009. Т. 4. С. 172—197.

23. **Осечкина М. С., Брыксин Т. А., Литвинов Ю. В., Кириленко Я. А.** Поддержка жестов мышью в Meta-CASE-системах // Системное программирование. 2010. Т. 5, № 1. С. 52—75.

24. **Терехов А. Н., Брыксин Т. А., Литвинов Ю. В.** QReal: платформа визуального предметно-ориентированного моделирования // Программная инженерия. 2013. № 6. С. 11—19.

25. **Terekhov A. N., Romanovskii K. Yu., Koznov D. V., Dolgov P. S., Ivanov A. N.** RTST++: Methodology and CASE Tool for the Development of Information Systems and Software for Real-Time systems // Programming and Computer Software. 1999. V. 25, N 5. P. 276—281.

26. **Терехов А. Н.** RTST — технология программирования встроенных систем реального времени. Записки семинара кафедры системного программирования "Case-средства RTST++". 1998. С. 3.

27. **Сухов А. О.** Инструментальные средства создания визуальных предметно-ориентированных языков моделирования // Фундаментальные исследования. 2013. № 4. С. 848—852.

28. **Кознов Д. В., Перегудов А. Ф., Бугайченко Д. Ю., Чернятчик Р. И., Казакова А. С., Павлинов А. А.** Визуальная среда проектирования систем телевизионного вещания // Системное программирование. 2006. Т. 2, № 1. С. 142—168.

29. **Кознов Д. В.** Предметно-ориентированное визуальное решение для сбора и упорядочивания информации при разработке информационной Web-системы // Компьютерные инструменты в образовании, 2013. Принято к печати.

30. **MEGA** Studio User Guide. 2-nd Edition (November 2011). MEGA International. 114 p.

31. **Каменова М., Громов А., Ферапонтов М., Шматалюк А.** Моделирование бизнеса. Методология ARIS. Практическое руководство. М.: Весть-Мета Технология, 2001. 333 с.

32. **CMap**. URL: http://cmap.ihmc.us/download/

33. **Иванов А. Н.** Технологическое решение Real-IT: создание информационных систем на основе визуального моделирования // Системное программирование. 2005. Т. 1. № 0. С. 89—100.

34. **Иванов А. Н., Стригун С. С.** Технологическое решение Real-IT: автоматизированная разработка пользовательского интерфейса информационных систем // Системное программирование. 2005. Т. 1, № 0. С. 124—147.

35. **Иванов А. Н.** Механизм поддержки циклической разработки ИС в рамках модельно-ориентированного подхода // Системное программирование. 2005. Т. 1, № 0. С. 101—123.

36. **Sendall S. and Küster J.** Taming Model Round-Trip Engineering // Proceedings of Workshop on Best Practices for Model-Driven Software Development. 2004. P. 1—10.

37. **Кознов Д. В., Иванов А. Н., Мишкинис А. И., Залевский Я. И.** Поддержка концептуального моделирования при разработке визуальных языков с использованием Microsoft DSL TOOLS // Системное программирование. 2009. Т. 4. С. 104—127.

38. **Костырко А., Кудрявцев Д., Григорьев Л., Кислова В., Жулин А., Синятуллина Л., Ермаков Р.** Моделирование комплексов городского хозяйства для системного развития ИКТ города // Сборник трудов конференции "Инженерия знаний и технологии семантического веба — 2012", 1—9 октября, 2012. Санкт-Петербург, Россия. СПб: СПбГУ ИТМО, 2012. С. 81—88.

39. **Холтыгина Н. А., Кознов Д. В.** Обзор реализации механизма циклической разработки диаграмм классов и программного кода в современных UML-средствах // Системное программирование. 2010. Т. 5, № 1. С. 76—94.

40. **Гаврилова Т. А., Лещева И. А., Кудрявцев Д. В.** Использование моделей инженерии знаний для подготовки специалистов в области информационных технологий // Системное программирование. 2012. Т. 7, №. 1. С. 90—105.

41. **The Mind Mapping** Software Blog. URL: http://mindmapping.typepad.com/

42. **Koznov D., Pliskin M.** Computer-Supported Collaborative Learning with Mind-Maps // Communications in Computer and Information Science. 2008. V. 17. CCIS. P. 478—489.

43. **Кознов Д. В., Ларчик Е. В., Плискин М. М., Артамонов Н. И.** О задаче слияния карт памяти (Mind Maps) при коллективной разработке // Программирование. 2011. Т. 37, № 6. С. 56—66.

44. **Ларчик Е. В., Кознов Д. В., Плискин М. М.** Реализация механизма слияния карт памяти (Mind Maps) в продукте Comapping // Системное программирование. 2011. Т. 6, № 1. С. 6—25.

45. **Business** Process Model and Notation (BPMN). Version 2.0. OMG. January 2011. 538 p.

46. **Терехов А. А., Терехов А. Н.** Computing Curricula: software engineering и российское образование // Открытые системы. СУБД. 2006. № 8. С. 61—66.

47. **Henderson-Sellers B., Ralyté J.** Situational Method Engineering: State-of-the-Art Review // Journal of Universal Computer Science. 2010. V. 16, N 3. P. 424—478.

48. **Кознов Д. В.** Методика обучения программной инженерии на основе карт памяти // Системное программирование. 2008. Т. 3, № 1. С. 121—140.

УДК 519.688

С. Ю. Болотова, аспирант, Воронежский государственный университет
e-mail: bolotova.svetlana@gmail.com

Реализация многопоточности в релевантном LP-выводе

Релевантный LP-вывод представляет собой эффективное средство для разработки, верификации и оптимизации продукционно-логических систем. В работе описывается программная реализация многопоточного LP-вывода. Она основана на распараллеливании этапов построения нескольких множеств фактов, необходимых при выводе. Приводятся псевдокоды некоторых алгоритмов, модель параллельных вычислений в виде графа, а также архитектура программного продукта.

Ключевые слова: *обратный вывод, релевантность, логические уравнения, параллелизм, многопоточность*

S. Yu. Bolotova

Multi-threaded Relevant LP-inference Implementation

Relevant LP-inference is a very effective tool used for development, research and optimization in the area of logic-production systems. This paper describes the computational algorithm implementation for the multi-threaded relevant LP-inference. It is based on the parallelizing some of the steps used to create a number of sets of facts that are necessary for the inference. The paper gives the pseudo codes for some of the algorithms, the graph-style computational model and the basic architecture overview of the final program based on the algorithm itself.

Keywords: *backward inference, relevant backward inference, logical equations, parallelism, multi-threading*

Введение

Продукционно-логические системы представляют важное направление исследований в области искусственного интеллекта. Их используют в теории обучающих систем, систем принятия решений, а также при разработке практических экспертных систем, охватывающих различные прикладные области, такие как медицина, проектирование, геологоразведка и др. [1]. В последние годы в ряде работ [2, 3] показан и исследован продукционный характер многих моделей в информатике, в том числе непосредственно не относящихся к области искусственного интеллекта. Кроме того, имея несложную структуру, системы продукционного типа могут привлекаться в качестве стартового "полигона" для создания и изучения методов управления знаниями, которые в дальнейшем применяют для построения более сложных современных систем.

Следует однако отметить, что при высокой практической востребованности продукционные системы довольно ресурсоемки. В частности, многие связанные с ними алгоритмы имеют экспоненциальную сложность. Кроме того, продукционные системы часто требуют интенсивного обмена с внешней памятью. Данное обстоятельство снижает их эффективность. Для многих приложений подобная ситуация некритична. Тем не менее, существуют предметные области, где необходимы высокая производительность и отклик в режиме реального времени. Снижение производительности оказывает существенное влияние и на уровень проводимых теоретических исследований, так как многие специалисты избегают работы с алгоритмами, требующими предельных объемов ресурсов [1]. Учитывая изложенное выше можно констатировать, что задача повышения производительности продукционных и подобных им систем имеет важное теоретическое и практическое значение.

Существуют несколько возможных способов увеличения скорости работы продукционных систем [1]. К их числу относятся следующие:

1) применение аппаратных средств высокой производительности;

2) разработка более совершенных алгоритмов и общей архитектуры системы;

3) использование параллельных вычислений.

Настоящая статья посвящена в основном третьему пути решения отмеченной задачи и описывает реализацию параллелизма в моделях продукционных систем в части релевантного LP-вывода [4, 5]. Представлена архитектура объектно-ориентированного класса LPStructure, инкапсулирующего наиболее важные свойства и методы, описанные в работе [4], включая нахождение логической редукции и решение продукционно-логических уравнений. Новый элемент в данной реализации — распараллеливание вычислений. Для программирования выбран язык C++ с привлечением библиотеки STL в среде MS Visual Studio. Для обеспечения повторного использования данного класса в других программных системах его интерфейс оформлен в виде динамической С-библиотеки LPStructure.dll.

Общее описание алгоритма

Рассматриваемая программная система реализует алгоритм параллельного релевантного обратного вывода, основанного на решении продукционно-логических уравнений. Для этой цели применяют математически обоснованные в работе [4] механизмы нахождения истинного прообраза и ускорения обратного вывода. В данной статье используются обозначения и определения, принятые в указанной работе, а также известная терминология продукционных систем [1].

LP-структурой (*Lattice Production Structure*) называется математическая решетка с дополнительно заданным на ней бинарным отношением, которое обладает рядом продукционно-логических свойств.

Техника релевантного вывода основана на решении уравнений в LP-структурах. Она ориентирована на минимизацию числа медленно выполняемых запросов к внешнему источнику (базе данных или интерактивному пользователю). Обратный вывод начинается с построения всех минимальных начальных прообразов в LP-структуре для атомов, соответствующих значениям объекта экспертизы (гипотезы). Далее в построенном множестве достаточно найти прообраз, который содержит лишь истинные факты, после чего можно сделать заключение о соответствующем значении объекта экспертизы. Эффективным в этом плане является приоритетный просмотр прообразов, содержащих значения наиболее "релевантных" объектов [5]. Отрицательный ответ на запрос исключает последующие запросы об элементах связанного подмножества прообразов. Кроме того, при LP-выводе предпочтение отдается тестированию множеств фактов минимальной мощности.

Однако при больших объемах баз знаний и их достаточно "глубокой" структуре процесс последовательного построения всех минимальных прообразов может потребовать чрезмерного объема вычислительных ресурсов. В связи с данным обстоятельством метод модифицирован. Параллельный релевантный LP-вывод предполагает одновременное построение набора начальных прообразов с их дальнейшим исследованием в разных потоках. Здесь и далее под "потоками" подразумеваем *threads* — потоки выполнения [6].

Рассмотрим вопрос нахождения истинного прообраза. Пусть даны конечная атомно-порожденная решетка F (атомами изображаются элементарные факты) и на ней бинарное отношение R (представляет совокупность продукционных правил). Отношение R — каноническое, т. е. задано множеством пар вида (A, a), где $A \in F$, a — атом в F.

Пусть выбран атом $b \in F$. Ему соответствует общее решение продукционно-логического уравнения с гипотезой в правой части — множество $\{X\}$ всех его минимальных начальных прообразов в логическом замыкании отношения R. Пусть на множестве атомов решетки частично определена булева функция *True* (функция истинности), которая способна доопределяться путем обращения к внешнему источнику информации. В моделируемой продукционной системе интерпретация этой функции такова:

- $True(x) = 1$, если соответствующий факт x содержится в рабочей памяти;
- $True(x) = 0$, если достоверно известно, что x не может содержаться в рабочей памяти;
- $True(x) = null$, если проверка x еще не проводилась.

Введем обозначения $T = \cup x_k$ ($True(x_k) = 1$); $F = \cup x_j$ ($True(x_j) = 0$). Необходимо найти такой элемент $X^0 \in \{X\}$, что $X^0 \subseteq T$ (если он существует). Основное требование — в процессе решения задачи обойтись как можно меньшим числом доопределений функции *True*.

Приведем алгоритм релевантного LP-вывода [5].

```
// Релевантный LP-вывод
X^0 = null
{X} = getPreImages(b)
while X^0 = null and {X} = ∅ do
    k = getRelevantIndex({X}, T)
    Ask(x_k)
    foreach X_j ∈ {X} do
      if X_j ⊆ T then
        X^0 = X_j
        break
      end
      if X_j ∩ F ≠ ∅ then {X} = {X}\X_j
    end
end
```

Функция *getPreImages*(b) решает продукционно-логическое уравнение с гипотезой b в правой части, т. е. строит множество $\{X\}$ всех *минимальных* начальных прообразов для атома b. Функция *Ask*(x) запрашивает внешний источник об истинности атома x и в соответствии с ответом модифицирует множества T и F (т. е. доопределяет функцию *True*). Функция *getRelevantIndex*($\{X\}$, T)

находит индекс k любого из наиболее релевантных и ранее не проверенных на истинность атомов, содержащихся в элементах текущего $\{X\}$. Она использует упомянутые ранее показатели релевантности начальных атомов в целях снижения числа вызовов функции $Ask(x_k)$.

Для решения уравнения исходное каноническое отношение R представляется в виде слоев [4], каждый из которых порождает не более одного решения. Слой содержит максимально возможный набор пар исходного отношения с уникальными правыми частями. Непосредственная реализация слоев привела бы к избыточному хранению пар, так как слои могут иметь большие пересечения. Для решения вопроса представления слоев поступим следующим образом.

Разобьем R на непересекающиеся подмножества, каждое из которых образовано парами вида (A, x_p) с одним и тем же атомом x_p в качестве правой части. Обозначим эти подмножества R^p соответственно их элементу x_p, $p \in P$, где P определяет подмножества пар отношения R. При реализации канонического отношения R для каждого x_p достаточно хранить лишь совокупность левых частей пар с правой частью x_p. Каждому из подмножеств R^p поставим в соответствие итератор — индексную переменную j_p, перебирающую собственное подмножество, останавливаясь на каждой паре ровно один раз. Таким образом, любой слой R_t в отношении R получается соответствующим набором значений итераторов j_p. Нахождение решения в отдельном слое сводится к получению списка начальных вершин графа, из которых достижима данная вершина (она соответствует атому правой части уравнения).

Работа с различными слоями организована независимо и параллельно. Первичный поток приложения создает новые потоки (они ограничены параметром *MaxThreads* — максимальное их число), передавая им пакет данных. Созданный поток находит решение продукционно-логического уравнения в отдельном слое. Получая начальную порцию прообразов, модуль LP-вывода сразу исследует их на предмет истинности, обращаясь при необходимости за фактами к внешнему источнику. Если при обработке очередного прообраза он не оказывается истинным, то вычисляется следующий прообраз. При этом запоминается информация об установленных на предыдущем шаге ложных фактах. На ее основе перед очередным процессом решения уравнения сужается множество актуальных правил. Это обстоятельство также ускоряет работу. По окончании работы потоки завершаются.

В случае, когда рабочих потоков слишком много, эффективность вычислений снижается. Интенсивное создание и завершение потоков с малым временем работы, а также большое число переключений их контекстов увеличивают объем ресурсов. По этой причине используется заранее создаваемый пул потоков [6]. Главный поток выбирает поток из пула и передает ему необходимые данные для обработки. Если число активных потоков меньше максимального, создается новый поток, который получает пакет данных на обработку. Если же число активных потоков достигает максимума, пакет ставится в очередь и ждет освобождения одного из потоков. В качестве механизма синхронизации используются критические секции, которые инициируются в процессе активизации потоков.

При разработке параллельных алгоритмов важно оценивать их эффективность, т. е. получаемое ускорение работы. С этой целью можно использовать модель вычислений в виде ациклического графа $G = (V, R)$, в котором множество операций, выполняемых в исследуемом алгоритме решения задачи, представляется как множество вершин $V = \{1, ..., |V|\}$, а информационные зависимости между операциями — в виде множества дуг R [7]. Дуга $r = (i, j)$ означает, что операция j использует результат выполнения другой операции i, а те операции алгоритма, между которыми нет пути, могут быть распараллелены. Возможное описание параллельного выполнения алгоритма нахождения решения в отдельном слое с помощью графа изображено на рис. 1.

Далее приведен алгоритм функции *getPreImages*(b). Обозначим множество слоев $\{R_i\}$ через R'. Как уже отмечалось ранее, решение в каждом слое вычисляется отдельным потоком. Максимальное число потоков в пуле ограничено параметром *MaxThreads*. Число активных потоков хранится в переменной *countUsedThreads*. В случае, если в пуле есть свободный поток, он запускается и в нем вызывается функция *FindEquationDesicion*(R_i), которая находит решение уравнения в очередном слое R_i.

```
// Нахождение решения уравнения на каждом слое в отдельном потоке
foreach R_i ∈ R' do in parallel
    if countUsedThreads < MaxThreads then
        BeginThread(); // Начать выполнение потока
        countUsedThreads++;
        FindEquationDesicion(R_i);
        ExitThread(); // Завершить поток
        countUsedThreads--;
    else
        Waiting(); // Ожидание освобождения потока
    end
end
```

При большом числе прообразов процесс выявления релевантных объектов весьма ресурсозатратен, поэтому он также распараллеливается. Для очередной порции фактов в разных потоках (аналогично ограниченных параметром *MaxThreads*) одновременно вычисляется число прообразов, которым принадлежат данные факты. При этом для факта, находящегося в максимальном числе прообразов, показатель релевантности увеличивается. Для синхронизации потоков используется механизм критических секций. На рис. 2 представлен граф вычислений для алгоритма нахождения релевантности.

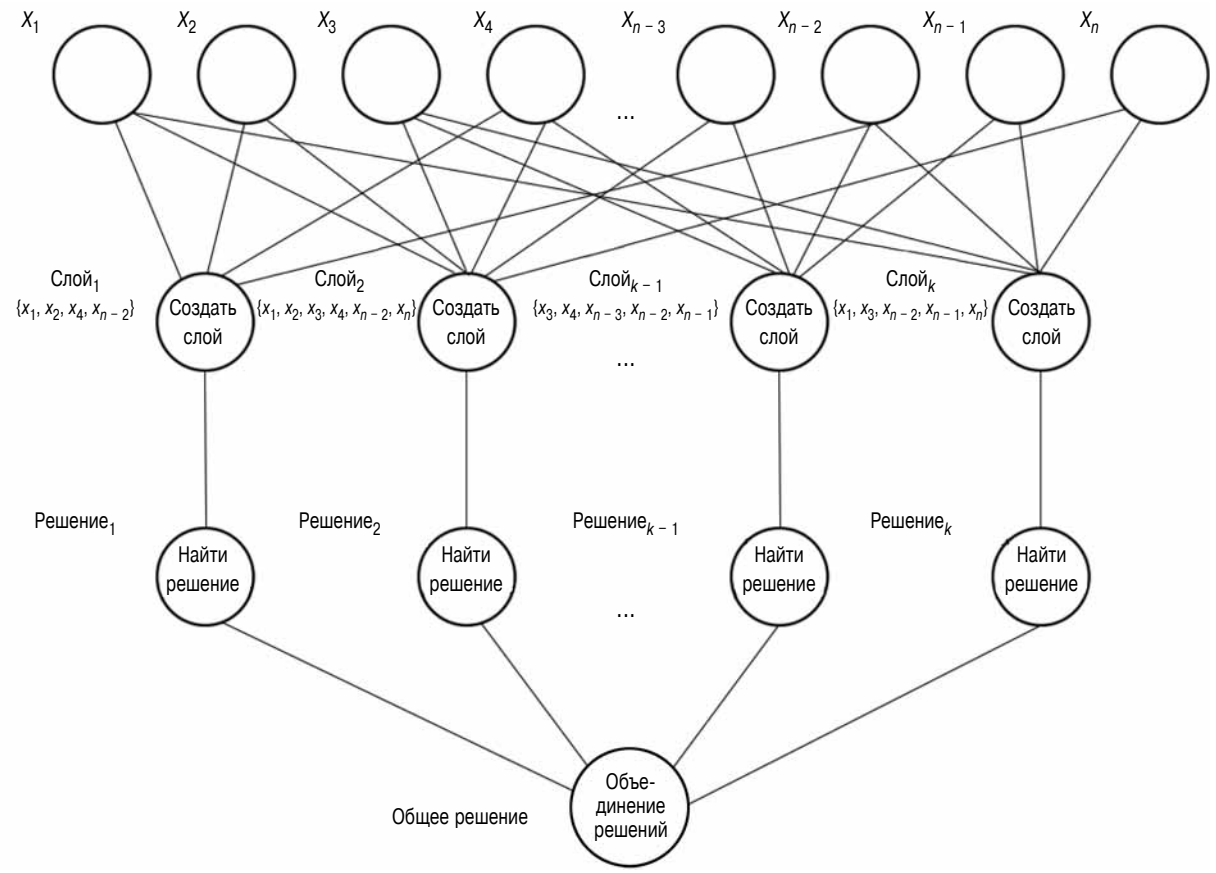


Рис. 1. Граф вычислений для параллельного алгоритма нахождения решения в отдельном слое

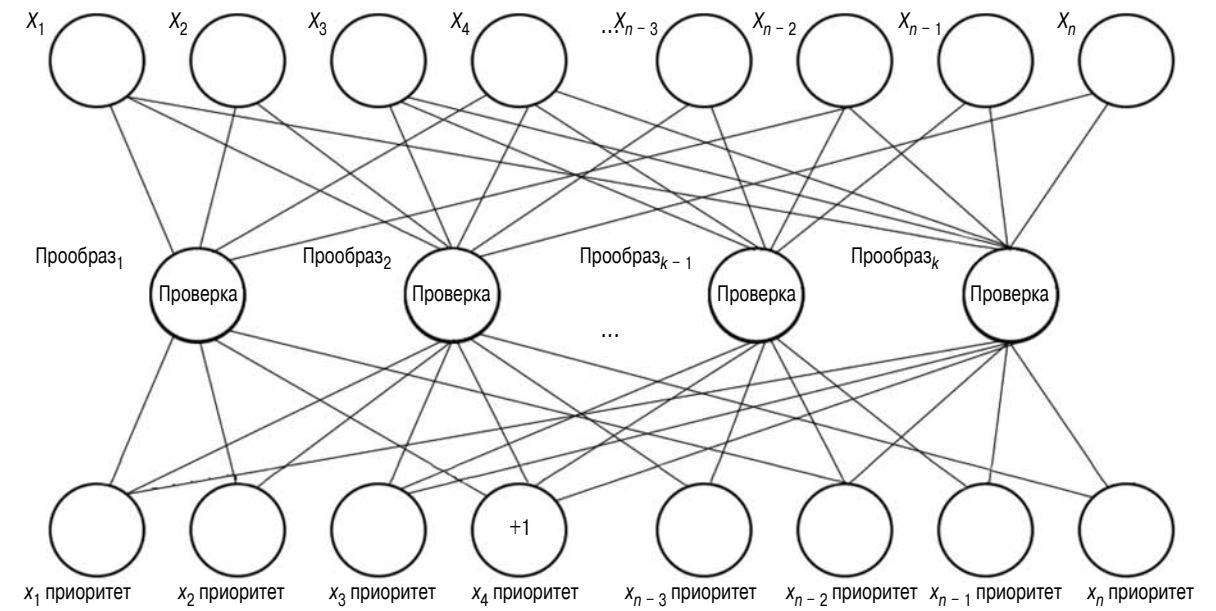


Рис. 2. Граф вычислений для алгоритма вычисления релевантности

Один из возможных вариантов функции *getRelevantIndex* представлен ниже.

```
// Подсчет релевантности
int getRelevantIndex ({X}, T)
    // Инициализация приоритетов начальных атомов решетки
    foreach x_k ∈ F_0 do Priority[k] = 0
    // Нахождение минимальной мощности прообразов
    int nMin = null;
    foreach X_j ∈ {X} do
       if nMin = null or Length(X_j) < nMin then nMin = Length(X_j)
    end
    // Вычисление приоритетов начальных атомов
    foreach x_k ∈ F_0 do in parallel
       if countUsedThreads < MaxThreads then
          BeginThread(); // Начать выполнение потока
          countUsedThreads++;
          if not x_k ⊆ T ∪ F then // Рассматриваются лишь непроверенные атомы
              foreach X_j ∈ {X} do
                 if x_k ⊆ X_j then Priority[k]++
              end
              ExitThread(); // Завершить поток
              countUsedThreads−−;
          end
       else
          Waiting(); // Ожидание освобождения потока
       end
       foreach X_j ∈ {X} do
          if Length(X_j) = nMin then Priority[k]++
       end
    end
    // Нахождение индекса наиболее релевантного атома
    int nRel = null;
    foreach x_k ∈ F_0 do
       if nRel = null or Priority[k] > Priority[nRel] then nRel = k
    end
    return nRel
end
```

Оптимальное число потоков устанавливается опытным путем. Как показали эксперименты, параллельный релевантный LP-вывод дает эффективные результаты для обработки баз знаний больших размеров.

Внутренний формат данных

Рассмотрим практические соображения, которые положены в основу программной реализации LP-структур. В соответствии с работой [5], для хранения элементов решетки используется гибкая конструкция битового вектора, при которой его длина может настраиваться клиентской программой. Размерность такого вектора задается двумя параметрами — длиной в байтах одного кластера (*eItemSize*) и числом этих кластеров (*eLengthItems*).

Значения *eItemSize* определяет размер кластера — части вектора, которая обрабатывается единственной логической операцией компьютерного процессора. Это значение может храниться, например, как статическое константное поле объектно-ориентированного класса LPStructure, и меняться с его перекомпиляцией. Выбором *eItemSize* = 1 можно структурировать битовый вектор в виде последовательности байтов, что приведет к экономии памяти и сделает структуру вектора более прозрачной для понимания алгоритмов. Если же положить, например, *eItemSize* = 4, то можно повысить быстродействие операций над векторами, так как многие циклы в программе сократятся в 4 раза, а 32-разрядные операции окажутся более эффективными для ЭВМ с соответствующей длиной слова памяти.

Число кластеров битового вектора хранится в целочисленном поле *eLengthItems*. Конструктор класса LPStructure получает в качестве параметра соответствующее целочисленное значение для настройки данного поля. Это обстоятельство предоставляет возможность клиентской программе при создании LP-структуры динамически настраивать длину битовых векторов, исходя из размера решетки и объема имеющейся базы знаний.

В классе LPStructure определены типы *Elem* и *Pair* путем переименования соответственно типов обычного и двойного целых чисел. Тип *Elem* предполагает хранение адреса элемента решетки, а именно адреса соответствующего элементу битового вектора. Тип *Pair* при этом содержит смежную пару величин *Elem*, т. е. предназначается для хранения пары элементов (точнее, их адресов) из некоторого бинарного отношения на решетке. Конечно, типы *Elem* и *Pair* можно реализовать в виде объектно-ориентированных классов с перегруженными операторами (|, &, < =), соответствующими операциям на решетке. Однако решение использовать простейшие типы оказывается более последовательным в плане принятой стратегии экономии памяти.

При использовании библиотеки STL для реализации LP-структур широко применяют ее эффективные параметризованные контейнерные классы *vector* и *set*, а также их комбинации. В частности, для представления множеств элементов решетки полезным оказывается тип *ElemContainer*, определяемый как *set*<*Elem*>. Для хранения задаваемых на решетке бинарных отношений общего вида определяется тип *PairContainer* как *set*<*Pair*>. Некоторыми алгоритмами в качестве промежуточных данных используются векторы пар — *PairVector* (*vector*<*Pair*>). Каноническое бинарное отношение, в соответствии с изложенными выше соображениями, представимо вектором множеств *ElemContainerVector* (*vector*<*ElemContainer*>).

Параметр *aMaxThread* ограничивает число рабочих потоков. Он задается как статическое константное поле класса LPStructure.

Базовые функции библиотеки

Класс LPStructure инкапсулирует перечисленные далее основные методы. Конструктор

LPStructure (**const int** *aLen* = 8);

в качестве параметра принимает значение целочисленного поля *eLengthItems* для настройки длины битовых векторов. Как видно из объявления, по умолчанию принята структура вектора в виде байтовых кластеров.

Ряд методов предназначен для обеспечения стандартных операций над множеством пар LP-структуры. К ним относятся следующие функции, назначение которых следует из их названий:

BOOL insert (**const** *Pair pair*);
BOOL insert (**const** *Elem left*, **const** *Elem right*);
Pair extract ();
BOOL erase (**const** *Pair pair*);
BOOL erase (**const** *Elem left*, **const** *Elem right*);
void *clear* ();
ULONGLONG count () **const**;

Несколько методов реализуют основную функциональность класса. Они имеют два общих параметра
const *OnLPEvent onEvent* = *NULL*,
const *DWORD dwUser* = 0.

Эти параметры обеспечивают возможность сообщения "клиенту" детальной информации о событиях, происходящих во время их работы. Параметр *onEvent* (если задан) содержит адрес функции обратного вызова — обработчика LP-событий. Параметр *dwUser*, как это принято во многих функциях WinAPI [6], содержит сопутствующую информацию "пользователя". Обработчик LP-событий обязан иметь следующий прототип:

typedef void (*__cdecl *OnLPEvent*) (**const** *Pair aPair*,
const int *nEventType*,
const *DWORD dwUser*);

Здесь параметр *aPair* содержит пару элементов решетки, связанную с произошедшим событием. Параметр *nEventType* передает код события, *dwUser* возвращает информацию "пользователя", которая может быть использована им по собственному усмотрению, например, для идентификации источников событий. Предусмотрены следующие виды событий (возможные значения параметра *nEventType*):

static const int *etRedundant* = 0; // Лишнее правило
static const int *etInference* = 1; // Элемент прямой
// логической связи
static const int *etPreImageCount* = 2; // Число прообразов
static const int *etPreImage* = 3; // Очередной прообраз

Следующие два метода предназначены для построения логической редукции LP-структуры [4] на основе логических связей пар элементов. Функция

BOOL isLConnected (**const** *Pair aPair*,
const *OnLPEvent onEvent* = NULL,
const *DWORD dwUser* = 0);

определяет наличие логической связи для пары элементов, задаваемой параметрами *aPair*. Остальные параметры описаны выше. Они позволяют "клиенту" получить структуру логической связи, если она будет обнаружена.

Функция

ULONGLONG lReductionLC
(**const** *OnLPEvent onEvent* = *NULL*,
const *DWORD dwUser* = 0);

строит логическую редукцию LP-структуры, обращаясь в процессе работы к функции *isLConnected*. Ее параметры позволяют при необходимости сообщить "клиенту" об обнаруженных "лишних" парах. В качестве результата она возвращает число пар редуцированной LP-структуры. Метод

void *maxImage* (**const** *Elem eSource*,
const *Elem eRes*, **const** *OnLPEvent onEvent* = *NULL*,
const *DWORD dwUser* = 0);

для заданного элемента решетки eSource вычисляет в LP-структуре его наибольший образ *eRes*. Параметры *onEvent* и *dwUser* позволяют "клиенту" получать ин-

формацию о парах, которые вносят вклад в прямой логический вывод. Метод

BOOL minPreImages (**const** *Elem eDest*,
const *Elem aNegative* = 0,
const int *aMaxCount* = 0,
const *OnLPEvent onEvent* = *NULL*,
const *DWORD dwUser* = 0, **const int** *nOptimize* = *otMemory*);

предназначен для нахождения решений продукционно-логических уравнений. Для заданного элемента решетки *eDest* он вычисляет минимальные прообразы, о которых сообщает вызывающей программе посредством параметров *onEvent* и *dwUser*. Параметр *nOptimize* задает один из двух возможных режимов оптимизации работы — по памяти или по времени.

Два специальных параметра функции *minPreImages* носят более прикладной характер, чем остальные. Параметр *aNegative*, если задан, содержит объединение начальных атомов решетки, которые должны быть принудительно исключены из процесса поиска прообразов. Указанные атомы могут, например, соответствовать заведомо ложным фактам базы знаний. Учет их наличия позволяет сократить время работы алгоритма вычисления прообразов. Если параметр *aMaxCount* не равен нулю, он содержит целочисленное значение, ограничивающее число искомых прообразов. Когда этот параметр положителен, при достижении указанного им числа найденных прообразов процесс прекращается, и найденные прообразы немедленно возвращаются клиенту. Отрицательное значение параметра *aMaxCount*, точнее, его абсолютная величина, интерпретируется как период времени — число миллисекунд, отведенных для вычисления прообразов.

Параметры *aNegative* и *aMaxCount* позволяют пошагово решать продукционно-логическое уравнение, получая и обрабатывая решения ограниченными порциями. Данный способ используется алгоритмом кластерно-релевантного LP-вывода. Функция

void *minPreImageInParallel* (**const** *Elem eDest*,
const *ElemContainer * layer*,
const *CallbackEvent callback*);

предназначена для нахождения минимального начального прообраза в слое *layer* для заданного атома решетки LP-структуры *eDest*. Результат работы возвращается в главный поток посредством функции обратного вызова, адрес которой передается в аргументе *callback*:

typedef void (* *CallbackEvent*) (**const** *ElemContainer result*);

Для параллельного нахождения всех минимальных начальных прообразов предназначена функция

void *threadedMinPreImages* (*const Elem eDest*);

Она для каждого из доступных слоев, получившихся в результате исходного разбиения, запускает процедуру *minPreImageInParallel* в отдельном потоке, передавая указатель на список аргументов для этой функции.

Для работы с потоками используются функции WinAPI.

Заключение

При создании больших баз знаний не всегда учитывается объем ресурсов, необходимый для их обработки. В результате процессы работы с базой знаний, включая само установление гипотезы, требуют большого объема вычислительных ресурсов компьютера. Усовершенствование алгоритмов логического вывода позволяет уменьшить такие затраты.

Представленная библиотека демонстрирует работоспособность теории LP-структур в целях ускорения обратного вывода в продукционных системах, а также программную оптимизацию, основанную на применении многопоточности. Описаны основные функциональные возможности класса LPStructure, что позволяет более точно судить и о принципах его построения.

Список литературы

1. **Gupta A.** Parallelism in production systems. Pitman, 1987. 224 p.

2. **Махортов С. Д.** LP-структуры на решетках типов и некоторые задачи рефакторинга // Программирование. 2009. Т. 35, № 4. С. 5—14.

3. **Махортов С. Д.** Основанный на решетках подход к исследованию и оптимизации множества правил условной системы переписывания термов // Интеллектуальные системы. 2009. Т. 13, вып. 1—4. С. 51—68.

4. **Махортов С. Д.** Логические уравнения на решетках // Вестник ВГУ. Серия Физика, математика. 2004, № 2. С. 170—178.

5. **Болотова С. Ю., Махортов С. Д.** Алгоритмы релевантного обратного вывода, основанные на решении продукционно-логических уравнений // Искусственный интеллект и принятие решений. 2011. № 2. С. 40—50.

6. **Рихтер Дж.** Windows для профессионалов: создание эффективных Win32-приложений с учетом специфики 64-разрядной версии Windows: пер. с англ. 4-е изд. СПб.: Питер; М.: "Русская Редакция", 2001. 732 с.

7. **Воеводин В. В., Воеводин Вл. В.** Параллельные вычисления. СПб.: БХВ-Петербург, 2002. 608 с.

УДК 004.056

А. А. Сирота, д-р техн. наук, проф., **К. А. Титов,** аспирант, Воронежский государственный университет,
e-mail: ktitovy@gmail.com

Архитектура распределенной информационной системы для поддержки технологий создания цифровых водяных знаков*

Предлагается и обосновывается архитектура распределенной информационной системы, позволяющая обеспечить поддержку технологии создания цифровых водяных знаков в интересах защиты авторских прав на медиа-контент.

Ключевые слова: *цифровой водяной знак, защита авторских прав, информационная безопасность, распределенная информационная система, медиа-контент*

A. A. Sirota, K. A. Titov

The Architecture of Distributed Information System to Support the Digital Watermarking Technology

Proposed and justified architecture of distributed information system to provide support for the digital watermark to protect copyright on a multimedia content.

Keywords: *digital watermark, copyright protection, information security, distributed information system, media content*

В настоящее время вопросы защиты авторских прав на объекты мультимедиа, такие как видео-, аудиоданные и изображения, занимают одно из центральных мест в сфере защиты информации. Причина в массовом распространении мобильных устройств, способных создавать, воспроизводить и распространять такой контент, в развитии ресурсных возможностей сети Интернет. Существуют различные подходы к решению задачи защиты авторских прав, которая носит комплексный характер. Эти подходы должны базироваться на реализации мероприятий организационного (юридические аспекты), организационно-технического и технологического характера. Одним из базовых подходов в области информационных технологий для защиты авторских прав на медиа-контент является применение технологий цифровых водяных знаков (ЦВЗ) [1, 2]. Цифровые водяные знаки представляют собой специальные метки, скрытно встраиваемые в медиа-файл, служащий контейнером — носителем соответствующей информации. Цифровые водяные знаки могут быть использованы в следующих целях: контроля копирования; доказательства права владения; выявления подделок; контроля версий; контроля используемых устройств; множественного использования [3]. Для того чтобы обеспечить защиту авторских прав и с учетом особенностей медиа-контейнера, цифровая метка должна удовлетворять следующим критериям: быть невидимой для человеческого глаза; не давать возможности удалять ее без существенных потерь качества контента; обладать стойкостью к основным искажениям контейнера; поддерживать низкую вероятность ошибки при извлечении цифровой метки [4, 5].

Существует большое число технологий встраивания ЦВЗ [6—9], которые в той или иной степени удовлетворяют описанным критериям. Однако как бы ни были хороши алгоритмы созданий и восстановления ЦВЗ, они не имеют практической ценности, если технологические решения в области ЦВЗ не имеют соответствующей организационно-технической поддержки. В контексте настоящей статьи имеется в виду,

* Исследование выполнено при финансовой поддержке РФФИ в рамках научного проекта № 13-01-97507 р_центр_а.

что для эффективной реализации таких решений необходимо использовать распределенную информационную систему (ИС), позволяющую регистрировать создание ЦВЗ, выявлять факт незаконного использования медиа-контента, прослеживать распространение его по сети Интернет в целом и между конечными пользователями. По этой причине целью настоящей работы является обоснование архитектуры подобной ИС, ориентированной на поддержку технологий ЦВЗ для защиты объектов, создаваемых и распространяемых с использованием мобильных устройств.

Требования к ИС поддержки технологии ЦВЗ

Требования к ИС определяются постановкой задачи и особенностями медиа-контента. Основные из них в кратком изложении сводятся к наличию механизмов, которые обеспечивают:

- независимость от конкретного алгоритма создания и формата ЦВЗ;
- возможность создания медиа-контента с выбором опции встраивания и восстановления ЦВЗ;
- доказательство права владения;
- невозможность отказа от авторства;
- контроль копирования медиа-контента;
- возможности контроля распространения медиа-контента.

Таким образом, основным требованием к системе является ее универсальный характер и независимость от технологии создания ЦВЗ, что обеспечивает возможность повышения качества встраивания меток без изменения архитектуры самой системы. Информационная система должна поддерживать профили пользователей и возможность создавать медиа-контент с ЦВЗ, ассоциированный с пользователем системы. Важными требованиями являются наличие в системе механизмов, которые не позволяют пользователю отказаться от авторства, а также возможность контроля распространения контента, наличие механизмов, позволяющих проследить за его перемещениями по сети Интернет и между конечными пользователями.

Общая архитектура распределенной ИС поддержки технологий ЦВЗ

В качестве типового решения предлагается ИС распределенной клиент-серверной архитектуры (рис. 1).

Она содержит три основные части:

- клиентская часть (мобильное приложение, предоставляющее пользователю основные функциональные возможности с точки зрения создания контента);
- серверная часть (серверное приложение, обрабатывающее запросы пользователей к необходимым данным);
- база данных (хранит информацию о пользователях, их контенте и его перемещениях по сети).

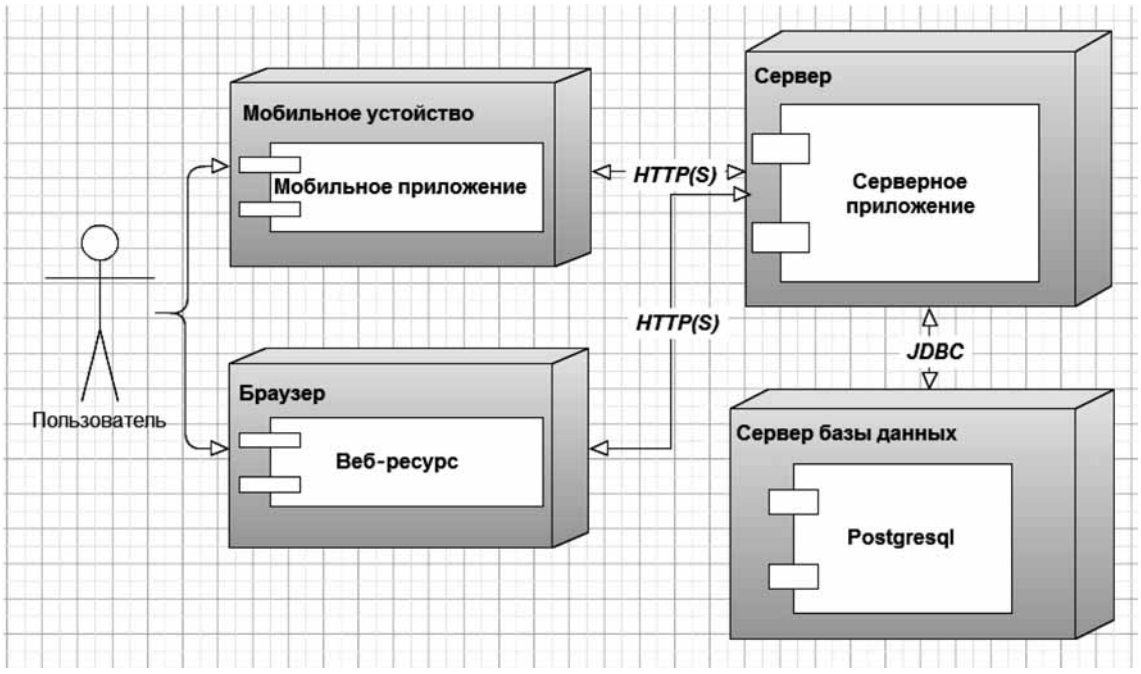


Рис. 1. Диаграмма развертывания системы

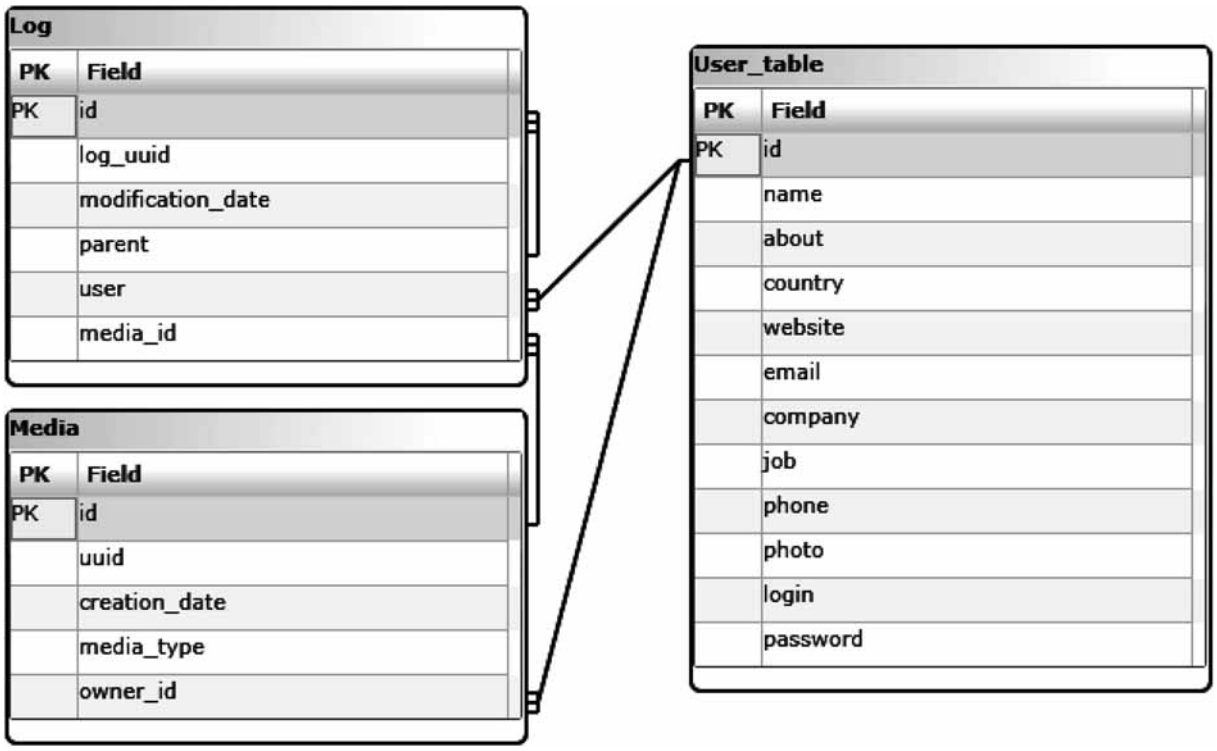


Рис. 2. Структуры базы данных (фрагмент)

Пользователь имеет доступ непосредственно к клиентской части. В предложенной программной реализации в качестве клиентской части выступает устройство, которое позволяет создавать или просматривать медиа-объекты, а именно: мобильный телефон; фотоаппарат; планшетный компьютер или персональный компьютер. Выбор именно этих устройств следует из требований к ИС, так как встраивание цифровой метки проводится только в момент создания видеоролика, аудиозаписи или съемки изображения. Такой подход позволяет сделать пользователя правообладателем контента без промежуточных действий и лишает возможности присвоения контента другим пользователем в силу отсутствия неподписанного медиа-контента. Предлагаемая реализация системы обеспечивает невозможность отказа от авторства ввиду отсутствия механизмов выхода из системы или очистки изображения от цифровой метки.

Серверная часть распределенной ИС поддерживает интерактивность и механизмы распределения прав доступа к данным, а также предоставляет клиентским приложениями данные. Для получения доступа к данным клиентское приложение должно пройти авторизацию или регистрацию посредством уникального идентификатора и пароля. Серверная часть играет роль удостоверяющего центра и служит для безопасного доступа к базе данных. Ввиду постоянной доступности серверное приложение обеспечивает необходимую интерактивность, объединяя клиентские приложения в единую распределенную систему.

База данных является единым хранилищем данных о медиа-контенте, пользователях и изменении контента. Доступ к базе данных осуществляется через серверное приложение. Фрагмент структуры базы данных представлен на рис. 2.

Здесь таблица User_table хранит информацию о профилях пользователей системы. Таблица Media содержит записи о медиа-контенте пользователей. Таблица Log содержит записи о фактах использования медиа-контента другими пользователями.

Функционирование ИС

Рассмотрим последовательность действий, выполняемых в процессе функционирования системы. Диаграмма деятельности приведена на рис. 3.

Пользователь при первом запуске приложения выбирает профиль, под которым желает быть идентифицирован. Если профиль еще не зарегистрирован в системе, то посылается запрос на сервер с требованием регистрации. Например, пользователь делает фотографию через камеру устройства, а в этот момент в устройство будет добавлен ЦВЗ и на сервер отправится информация о метке, изображении и владельце. Вла-

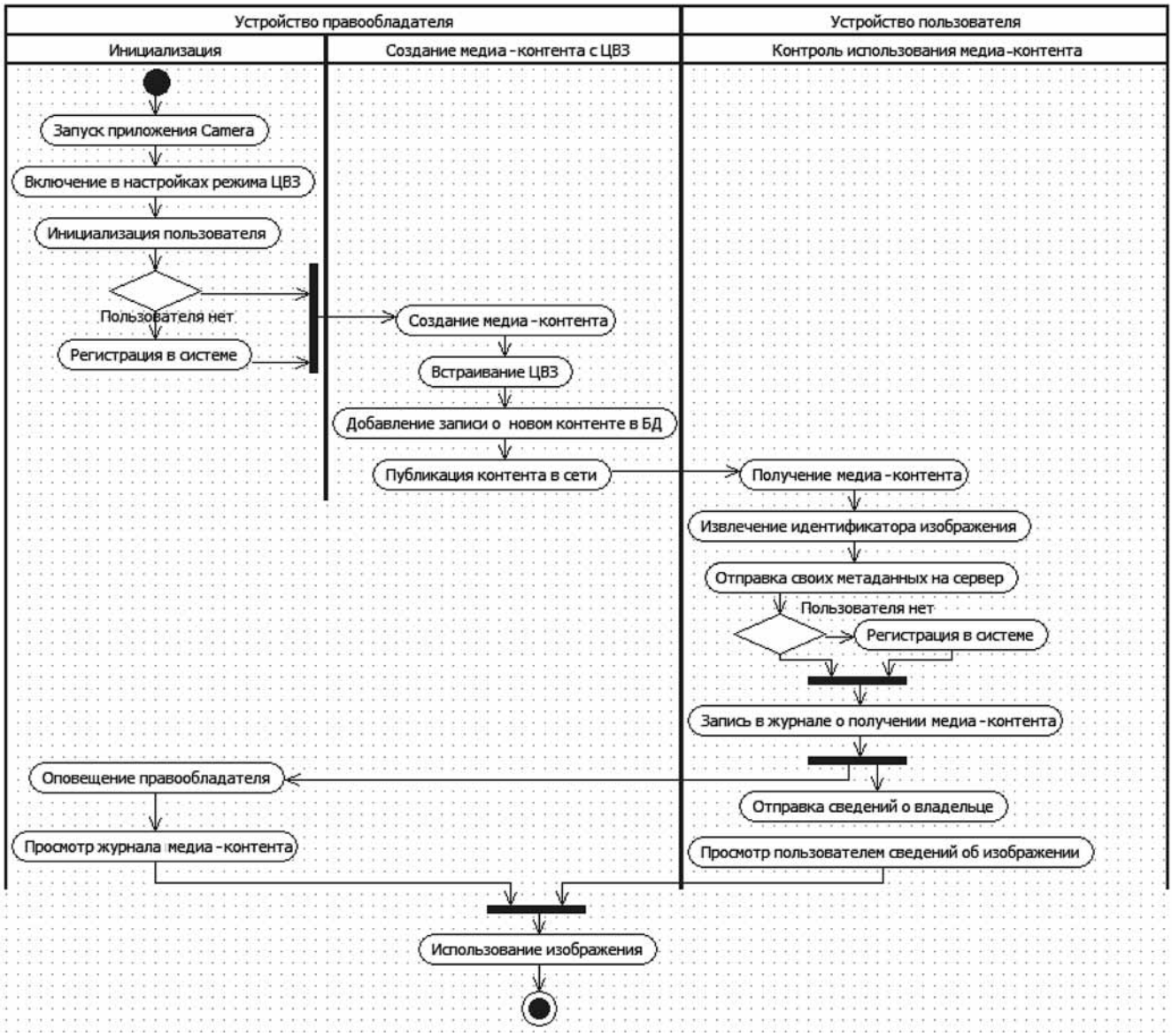


Рис. 3. Диаграмма деятельности системы

делец публикует изображение, и оно попадает к другому пользователю. Пользователь открывает изображение посредством приложения системы. В момент показа изображения приложение проверяет наличие встроенной метки, принадлежность ее пользователю, и если тот не является владельцем, на сервер будет отправлена запись о факте использования. Владелец изображения через журнал использования может обнаружить факт использования медиа-контента, а пользователь имеет возможность узнать информацию о владельце контента. Таким образом обеспечивается контроль распространения медиа-контента в ИС.

Программная реализация

В качестве одного прототипа из возможных был разработан демонстрационный вариант ИС. Клиентская часть в ней представляет собой программу, реализованную на платформе Android, а серверное приложение написано на языке Java, Play Framework [10]. В качестве базы данных выбрана PostgreSQL. Разработка мобильного приложения проводилась на языке Java в среде Eclipse. Приложение поддерживает возможность работы в портретной и ландшафтной ориентациях экрана. Локализация приложения предусматривает русскую и английскую версии интерфейса. Версии поддерживаемых Android устройств —

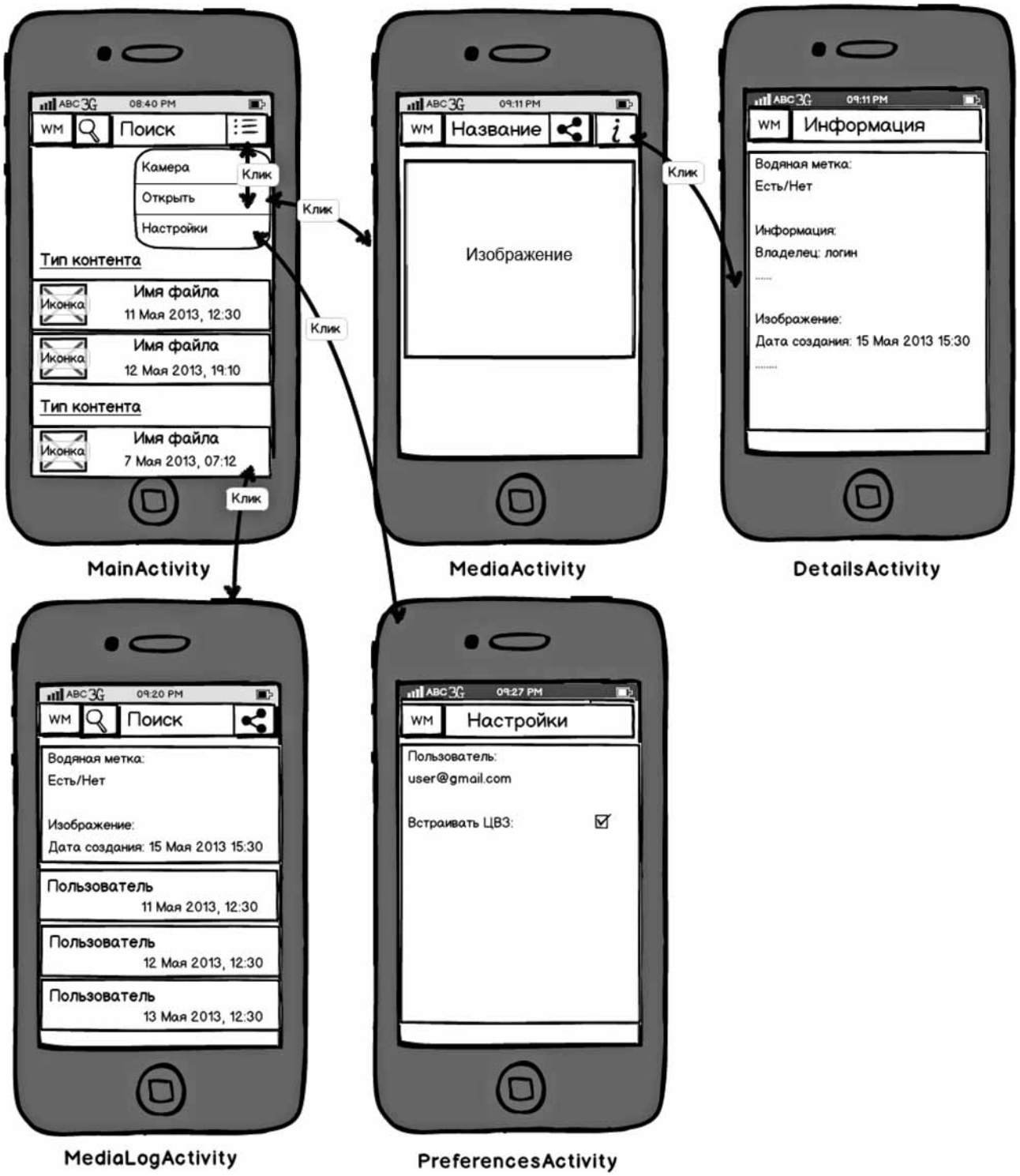


Рис. 4. Схема экранов Android-приложения

не ниже Android SDK 4.0.3. Схема экранов приложения и переходов между ними представлена на рис. 4.

На экране MainActivity отображается список медиа-контента пользователя. Пользователь может установить необходимость встраивания метки через настройки, а посредством платформенной камеры сделать снимок или видеозапись, в которую будет добавлена метка. Возможна публикация контента через социальные сети или через электронную почту. На экране MediaActivity можно увидеть медиа-контент, а также узнать информацию о пользователе и цифровой метке на экране DetailsActivity. Экран MediaLogActivity содержит список просмотров изображения другими пользователями, что позволяет следить за распространением контента в системе.

Проведено функциональное тестирование клиентской и серверной частей разработанной ИС по следующим тестовым сценариям: получение и отображение медиа-контента пользователя, встраивание цифровой водяной метки в мобильном приложении, отправка отчетов на сервер об использовании медиа-контента, а также сохранение и обработка данных о пользователях в базе данных.

Варианты применения ИС

Описанная ИС потенциально способна обеспечить комплексный контроль над распространением и копированием контента между пользователями только в том случае, если каждое приложение, используемое для воспроизведения медиа-контента в системе, будет способно поддерживать механизмы оповещения сервера о фактах использования контента. Такой подход может быть реализован в рамках корпоративных ИС или во встроенных сервисах мобильных устройств нового поколения. Возможно также использование предлагаемого решения в составе существующих сервисов сети Интернет, таких как сервисы компаний Google, Yandex и др. Пользователи, принимая пользовательское соглашение этих сервисов, будут создавать помеченный медиа-контент, а также им представится возможность следить за распространением своего медиа-контента. Такой подход, однако, не позволит контролировать глобальное перемещение контента за рамками сервисов. Причина в том, что для поддержки такого контроля программное обеспечение, воспроизводящее медиа-контент, должно реализовать функции оповещения сервера системы о фактах его использования. В связи с этим разработанный прототип информационной системы рассматривается как первый шаг на пути реализации организационно-технических мероприятий для поддержки технологий цифровых водяных знаков, которые способны обеспечить комплексную защиту авторских прав на медиа-контент.

Список литературы

1. **Мельников Ю. Н., Теренин А. В., Погуляев В. Г.** Цифровые водяные знаки — новые методы защиты информации // PC Week/RE. 2007. № 48 (606). URL: http://www.pcweek.ru/security/article/detail.php?ID=105054.

2. **Барсуков В. С., Шувалов А. В.** Еще раз о стенографии — самой современной из древнейших наук // Специальная техника. 2004. № 2. URL: http://www.ess.ru/sites/default/files/files/articles/2004/02/2004_02_04.pdf

3. **Грибунин В. Г., Оков И. Н., Туринцев И.В.** Цифровая стеганография. М.: Солон-пресс, 2002. 272 с.

4. **Бахрушин А. П.** Спектральный анализ видеокадров на основе системы импульсных функций с целью синхронизации процессов внедрения и поиск цифровых водяных знаков // Вестник ТОГУ. 2008. № 4 (11). С. 225—238.

5. **Петраков А. В., Лагутин В. С.** Защита абонентского телетрафика. М.: Радио и связь, 2001. 504 с.

6. **Хорошко В. А., Чекатков А. А.** Методы и средства защиты информации. Киев: ЮНИОР, 2003. 504 с.

7. **Левиев Д. А.** Описание алгоритмов встраивания ЦВЗ. Материалы курса "Защита информации" кафедры радиотехники Московского физико-технического института (МФТИ). URL: http://re.mipt.ru/infsec/2004/essay/2004_Watermark_algorithms_and_stability__LevievD.pdf.

8. **Дрюченко М. А., Сирота А. А.** Нейросетевые модели и алгоритмы стеганографического скрытия информации // Труды Российского научно-технического общества радиотехники, электроники и связи имени А. С. Попова. Москва, 2010. Т. 2. С. 335—338.

9. **Сирота А. А., Дрюченко М. А.** Нейросетевые модели и алгоритмы стеганографического скрытия информации // Информационные технологии. 2011. № 3. С. 41—49.

10. **Митекин В. А., Сергеев А. В., Федосеев В. А., Богомолов Д. М.** Построение модели стеганографической системы и обобщенного алгоритма встраивания ЦВЗ в полиграфические изделия // Компьютерная оптика. 2007. Т. 31, № 4. С. 95—101.

УДК 004.624, 025.3/.6

С. А. Афонин, канд. физ.-мат. наук, вед. науч. сотр., НИИ механики МГУ имени М. В. Ломоносова, e-mail: serg@msu.ru, **А. Э. Гаспорянц,** студент, МГУ имени М. В. Ломоносова

Разрешение неоднозначности авторства публикаций при автоматической обработке библиографических данных

Рассматривается задача автоматического разрешения неоднозначности при определении авторства публикации в процессе добавления записей в библиографическую информационную систему. Описываются формальные критерии, позволяющие оценить качество определения авторов, которые основываются на анализе истории совместных публикаций. Тестирование, проведенное на реальных массивах данных, показывает, что точность определения авторов составляет более 92 %.

Ключевые слова: *библиографическая запись, автоматическая обработка, авторство, неоднозначность*

S. A. Afonin, A. E. Gaspariants

Scientific Article Authorship Disambiguation for Automated Bibliographic Records Processing

In this paper the problem of automatic authorship disambiguation is considered. A number of formal parameters that take into account information about previous co-authorship are introduced. These parameters form the basis for disambiguation procedure. Experimental results conducted on real data show that algorithm precision is as high as 92 %.

Keywords: *bibliographic record, automated processing, authorship, disambiguation*

Введение

Используемые на практике методы количественной оценки результатов научно-исследовательских работ отдельных ученых или организаций в настоящее время основываются на показателях цитирования научных статей. В основе такой оценки следующий принцип: чем больше ссылаются на работы конкретного ученого, тем более успешной можно считать его деятельность. Получение универсальной шкалы, позволяющей сравнивать эффективность работы ученых из различных областей, связано со значительными трудностями [1]. Например, число научных статей, которые ежегодно публикуются в областях математики и биологии, приблизительно совпадает, но число ссылок между работами в области биологии в 10 раз больше (список источников биологической статьи в среднем в 10 раз длиннее, чем у математической работы), а значит абсолютные значения показателей цитирования у математиков и биологов будут различаться. Могут быть существенные различия и в рамках одной научной дисциплины, связанные с "популярностью" того или иного направления исследований. Тем не менее показатели цитирования рассматриваются как один из основных методов объективной оценки результативности научной деятельности [2].

Для получения показателей цитирования конкретных статей могут использоваться специализированные базы данных, такие как Web of Science, Scopus или РИНЦ. Эти системы достаточно точно определяют общее число ссылок на конкретную статью — ошибки определения статей при анализе списка литературы минимальны. Однако вычисление показателей цитирования конкретного ученого осуществляется с большой погрешностью ввиду наличия однофамильцев и неточного определения авторства статей.

Например, если зарегистрировать персональный профиль в системе GoogleScholar с достаточно распространенной фамилией, то с высокой вероятностью он будет содержать статьи однофамильцев, и показатели цитирования или индекс Хирша будут существенно отличаться от реальных.

В настоящей работе рассматривается задача автоматического определения авторов статьи по ее библиографическим данным. Предполагается, что существует система, например, описанная в работе [3], содержащая список ученых с некоторыми из их публикаций, для которых выполнено разрешение конфликтов авторства. Требуется разработать алгоритм, который позволит определять авторов новых публикаций, поступающих в систему. Публикации описываются заголовком, списком авторов, названием журнала и другими библиографическими параметрами. Данные, которые уже находятся в системе, считаются корректными, т. е. все однофамильцы разделены и работы корректно приписаны соответствующим пользователям. В дальнейшем будем использовать понятие *пользователь* для обозначения записи в базе данных, которая соответствует конкретному ученому, а понятие *автор* — для обозначения строки, указанной в поле авторов новой библиографической записи. Определение авторства — это процесс сопоставления автора и пользователя. Например, одним из авторов публикации может быть указан "Иванов И.", а в системе могут быть пользователи с именами "Иванов Игорь" и "Иванов Иван". Задача алгоритма — определить, кто из этих пользователей является автором данной работы. При определении авторства могут учитываться различные параметры, в частности, текстовая близость между фамилией пользователя и автора, наличие других совместных работ у выбранных пользователей, тематическая близость новой работы и предыдущих работ пользователя и др.

Характерный сценарий применения алгоритма состоит в следующем. Текущее состояние базы данных используется в качестве обучающей выборки для алгоритма определения авторства. При добавлении новой работы, содержащей n авторов, алгоритм возвращает n рекомендаций. Каждому автору (напомним, что автор — это просто строка с его именем) может быть поставлен в соответствие либо идентификатор некоторого пользователя, либо специальное значение, означающее что автоматический выбор сделать не удалось. Полученные данные могут использоваться для информирования соответствующих пользователей системы о добавлении в систему новой работы. Следует отметить, что значительный практический интерес представляет смежная задача кластеризации публикаций. В рассматриваемом сценарии имеется большая база библиографических записей, но никакого предварительного определения авторов не проводилось.

Описание модели

При добавлении новой работы, содержащей N авторов $(a_1, ..., a_N)$, для каждого автора a_i составляется список X_i "похожих" пользователей системы. Такой список фактически является множеством кандидатов для авторства работы и может содержать идентификаторы всех пользователей, фамилии которых совпадают с фамилиями авторов или являются их транслитерацией. Далее необходимо из каждого списка выбрать не более одного элемента таким образом, что значение некоторой функции качества F будет максимальным по всем возможным наборам.

Пусть K — число пользователей системы. Каждой строке a, содержащей имя автора, можно поставить в соответствие некоторую функцию $\mathrm{sim}_a: 1, ..., K \to [0, 1]$, которая моделирует работу алгоритма поиска пользователей, похожих на автора a. Пусть $S = \{f: 1, ..., K \to [0, 1]\}$ есть множество всех таких функций, а P — множество конечных подмножеств S.

Оценка достоверности того, что пользователи $I = (i_1, ..., i_N) \in 2^K$ написали статью, авторами которой указаны $(a_1, ..., a_N)$, определяется функцией $F: P \times 2^K \to [0, 1]$. Первый аргумент F отражает результат применения алгоритма поиска похожих пользователей, а второй — сделанный выбор пользователей. Задача определения авторства состоит в нахождении такого набора I^0, при котором значение F максимально:

$$I^0 = \mathrm{argmax}_{I \in 2^K} F(\langle \mathrm{sim}_{a_1}, ..., \mathrm{sim}_{a_N} \rangle, \langle i_1, ..., i_N \rangle).$$

Заметим, что первый аргумент F фиксирован и зависит только от входных данных.

При реализации указанного метода определения авторства возникают следующие задачи. Во-первых, необходимо сформировать множество параметров, влияющих на значение функции F, и дать ее формальное определение. Во-вторых, необходимо реализовать эффективный алгоритм решения оптимизационной задачи. Основная цель данной работы состоит в формализации параметров функции F и ее определение.

Формализация критериев оценки

Пусть задан набор $\langle a_1, ..., a_N \rangle$ авторов и набор $\langle v_1, ..., v_N \rangle$, $v_j \in X_j$ пользователей. Целью данного раздела работы является построение формальных критериев оценки качества набора пользователей с точки зрения их возможного соответствия заданным авторам. Эти критерии в дальнейшем будут использоваться для построения функции F.

Пусть задан набор $\langle v_1, ..., v_N \rangle$, $v_j \in X_j$, определяющий конкретный выбор пользователей системы по заданному набору N авторов. Построим по этому набору гиперграф $H = (V, E)$, где $V = \{v_1, ..., v_N\}$ — множество

вершин, а $E = \{e_1, ..., e_m\}$ — множество ребер. Наличие гиперребра означает, что ученые, соответствующие входящим в состав гиперребра вершинам, имеют совместные публикации. Таким образом H отражает историю совместных публикаций пользователей V. Каждому гиперребру u графа H можно поставить в соответствие множество $B(u)$ всех совместных публикаций пользователей, которые входят в u. Без ограничения общности можно считать, что публикации определяются своим идентификатором и $B(e) \subset \mathbb{N}$, где $\mathbb{N}$ — множество натуральных чисел. Введем обозначения $E' = \{e \in E : |e| = 1\}$, $V' = V \backslash E'$, $N' = |V'|$, где $|e|$ обозначает степень гиперребра (мощность множества e). Для любого $e \in E$ выполнено: $\nexists u \in 2^V : e \subset u \wedge B(e) = B(u)$, что равносильно следующему условию: $\nexists v \in V : B(e) = B(e \cup v)$. Для каждого $e \in E$ определена функция $P(e) = |B(e)|$. Далее из всех ребер удалим вложенные в них подребра и все ребра $e: |e| = 1$, т. е. выполнено: $\forall e \in E \nexists e' \in E : e' \subset e$ и $\forall e \in E : |e| \neq 1$. Полученное множество ребер опять обозначим через E.

Каждая вершина v графа H однозначно соответствует автору $a(v)$ работы. Для удобства введем обозначение $p(v) = \mathrm{sim}_{a(v)}(v)$, т. е. $p(v)$ определяет степень "похожести" пользователя v соответствующему автору.

Выделим перечисленные ниже параметры гиперграфа, которые будут влиять на значение функции F.

Ранг гиперграфа $r(H)$ (максимальная степень ребра). Это максимальное число авторов, имеющих общие статьи. Следовательно, чем ближе $r(H)$ к N, тем больше значение F. Пусть максимум достигается на ребре g, тогда положим $C = P(g)$.

Параметр $\gamma(H) = |e_1 \cap e_2 \cap ... \cap e_m|$ характеризует, сколько авторов имеют соавторство со всеми авторами из V. Чем больше $\gamma(H)$, тем больше значение F.

Число компонент связности H — $k(H)$. Нахождение подмножества авторов в одной компоненте связности гиперграфа показывает, что они могли писать статьи в соавторстве. Чем меньше $k(H)$, тем больше значение F.

Назовем *трансверсальным* множество $T \subset V'$, если $\forall e \in E$, $T \cap e \neq \varnothing$. Числом трансверсальности $\tau(H)$ графа H назовем мощность минимального по мощности трансверсального множества.

Пусть $TT = TT(H)$ — семейство всех трансверсальных множеств $T \subset V$: мощность T равна $\tau(H)$. Каждое из этих множеств является "базисом" гиперграфа H, так как если $T = \{v_1, ..., v_k\}$, $k = \tau(H)$, то положив A_j равным множеству соавторов для v_j, т. е. $A_j = v \in V : \exists e \in Ev$, $v_j \subseteq e$, получаем, что $V' = A_1 \cup A_2 \cup ... \cup A_k$. T является минимальным набором с таким свойством. Для каждого $T \in TT(H)$ аналогично строим соответствующее ему множество A_j и вычисляем следующие функции:

$f(T) = \max 1 \leqslant j \leqslant k : A_{i_1} \cap A_{i_2} \cap ... \cap A_{i_k} \neq \varnothing$, где $(i_1, ..., i_k)$ является перестановкой чисел 1, ..., k;

$$g(T) = \max_{1 \leqslant j \leqslant k} |A_j|;$$

$$\beta_1(T) = \sum_{j=1}^{k} p(a_j);$$

$$\beta_2(T) = \max_{1 \leqslant j \leqslant k} p(a_j);$$

$$\beta_3(T) = \min_{1 \leqslant j \leqslant k} p(a_j).$$

Функция $f(T)$ вычисляет максимальное число пользователей, у которых есть общий соавтор.

Верны следующие соотношения:

$$1 \leqslant f(T) \leqslant \tau(H),\ 1 \leqslant g(T) \leqslant N',$$
$$0 < \beta_1 \leqslant \tau(H),\ 0 < \beta_3 \leqslant \beta_2 \leqslant 1.$$

Функция $f(T)$ показывает, насколько "близки" базисные наборы A_j. Заметим, что если $f(T) = \tau(H)$, то $k(H) = 1$, так как каждое A_j является связным множеством гиперграфа (пересечение всех ребер, входящих в A_j, непусто, оно содержит a_j по построению). Если $f(T) < \tau(H)$, то гиперграф может быть несвязным. Значение $g(T)$ указывает максимальное число соавторов у "базисных" авторов. Значения параметров β_1, β_2, β_3 показывают, насколько базисные авторы похожи на авторов поступившей статьи.

Определим функцию $\Phi(H)$ следующим образом: $\Phi(H) = \frac{1}{TT(H)} \sum_{T \in TT(H)} \frac{f(T)}{N'} \frac{f(T)}{\tau(H)} \exp(\beta_2(T) - 1) \times \times \frac{\beta_3(T)}{\beta_2(T)} \frac{\beta_1(T)}{\tau(T)}$. Функция $\Phi(H)$ принимает значения в полуинтервале $(0, +\infty)$, ее максимум достигается в "идеальном" случае: $g(T) = N'$ (среди авторов статьи есть пользователь, который ранее уже являлся соавтором для каждого другого пользователя из рассматриваемого набора), $f(T) = \tau(H)$, $\beta_1 = \tau(H)$ (имена всех пользователей в точности соответствуют именам авторов). Из последнего условия непосредственно следует, что $\beta_1 = \beta_2 = 1$.

Назовем *независимым* множество $A \subset V'$, если: $\forall u, v \in A \nexists e \in E : u, v \in e$. Это означает, что A — множество авторов, попарно не имеющих общих статей. Пусть M — мощность максимального по мощности независимого множества.

Пусть $A_H = A_H(H)$ есть множество всех независимых множеств графа H. Положим $q(A)$ равным единице, если независимое множество A лежит в одной

компоненте связности H, и нулю в обратном случае. Тогда положим

$$Q(A_H) = \frac{\sum_{A \in A_H} q(A)}{|A_H|}.$$

Определим функцию

$$\alpha(H) = \max_{e \in E} P(e)\left(\frac{1}{|e|}\sum_{v \in e} p(v)\right).$$

Эта функция задается только "весовыми" значениями вершин гиперграфа и ребер, такими как число общих статей и степень похожести авторов из базы на авторов из поступившей статьи.

Определение функции *F*

Пусть $I_{\gamma(H)}$ принимает значение 1, если $\gamma(H) = 0$, и значение 0 иначе. Функцию F определим следующим образом: $F(H) = \frac{1}{|k(H)|}(Cr(H)\exp(\gamma(H)) + I_{\gamma(H)}\Phi(H)) \times (1 + Q(A_H))\alpha(H)$. Очевидно, данная функция удовлетворяет сформулированным ранее условиям. Она принимает значения в полуинтервале $(0, +\infty)$, причем в "идеальном" случае $F(H) = C^2N\exp(N)$. Далее монотонно и взаимо-однозначно отображаем F на интервал (0, 1) с помощью функции $y(t) = \frac{t}{1+t}$. Чем ближе полученная функция $y(F(H))$ к 1, тем лучше соответствующий ей набор авторов.

Реализация и тестирование

Описанная функция F оценки качества выбора пользователей по заданному набору авторов публикации была реализована в виде программы на языке C++. На вход данной функции подаем набор строк — авторы статьи.

- **Шаг 1.** Для каждого автора, выполняя запрос к базе данных, строим набор, состоящий из id похожих авторов, упорядоченный по степени похожести.
- **Шаг 2.** Для каждого id из всех наборов, выполняя запрос к базе данных, строим множество статей, привязанных к данному автору.
- **Шаг 3.** Выбирая из каждого набора похожих авторов по одному представителю, формируем множество авторов, на котором вычисляем функцию $y(F)$.
- **Шаг 4.** Упорядочиваем все возможные комбинации авторов по значению функции $y(F)$.

Алгоритм программно реализован на основе стандартной библиотеки STL (для работы с множествами и списками) и библиотеки OTL версии 4.0 (для выполнения запросов к базе данных).

Результаты тестирования на реальных данных

Число авторов	Всего статей	Доля распознанных авторов
2	980	0,91
3	745	0,94
4	392	0,94
5	244	0,92
6	128	0,92
7	249	0,92
8	183	0,91
9	86	0,92
10	46	0,91

Проводились следующие тесты на основе данных системы ИСТИНА. Выбиралась статья из базы данных, для которой известен результат. Результат работы алгоритма сравнивался с правильным ответом. Сравнение проводилось усредненное — доля правильно распознанных авторов среди всех. Результаты тестирования представлены в таблице.

Заключение

В работе предлагается формальный метод оценки степени достоверности разрешения неоднозначности авторства публикаций. Представленные результаты тестирования программной реализации предлагаемого алгоритма показывают, что на реальных данных достигается точность более 92 % определения авторов работы. Направления дальнейших исследований могут включать разработку более эффективных алгоритмов минимизации целевой функции F, расширение множества используемых критериев за счет анализа выходных данных журналов, добавление элементов семантического анализа названий работ и их аннотаций.

Список литературы

1. **Pudovkin A. I., Garfield E.** Rank-normalized impact factor: A way to compare journal performance across subject categories // Proceedings of the American Society for Information Science and Technology. 2004. N 41 (1) P. 507—515.

2. **Hirsch J. E.** An index to quantify an individual's scientific research output that takes into account the effect of multiple coauthorship // Scientometrics. 2010. N 85 (3). P. 741—754.

3. **Васенин В. А., Афонин С. А., Голомазов Д. Д., Козицын. А. С.** Интеллектуальная Система Тематического Исследования Научной информации (ИСТИНА) // Информационное общество. 2013. № 1—2. С. 21—36.

УДК 004.[056.53 + 451.5 + 514]

Д. В. Ефанов, канд. техн. наук, доц., **П. Г. Рощин**, аспирант, НИЯУ "МИФИ", г. Москва,
e-mail: d@efanov.pro

Вопросы реализации мандатной модели многоуровневого разграничения доступа в графической системе

Дан анализ подходов к реализации мандатной модели многоуровневого доступа в операционной системе, в состав которой входит многооконная графическая система. Основой для графического интерфейса в операционных системах UNIX и Linux является система X Window. Для X Window приводится перечень практических вопросов, которые необходимо решить разработчикам защищенной операционной системы, включая визуализацию меток, доверенные X-клиенты и веб-браузер.

Ключевые слова: *мандатное управление доступом, MAC, MLS, linux, X Window, selinux, графический интерфейс, веб-браузер*

D. V. Efanov, P. G. Roschin

Implementation Issues of Mandatory Multilevel Access Control Model in Graphical System

In article implementation issues of mandatory multilevel access control model in an operating system which includes multiwindow graphic system is considered. Basis for the graphic interface in the UNIX and Linux operating systems is the X Window system. For X Window the list of practical issues is given which developers of an trusted operating system need to solve, including visualization of the labels, the trusted X-clients and the web-browser.

Keywords: *mandatory access control, MAC, MLS, linux, X Window, selinux, graphic interface, web-browser*

Постановка задачи

В соответствии с существующей нормативной базой [1, 2] в операционной системе (ОС), которая используется для обработки информации, составляющей государственную тайну, должен быть программно реализован "мандатный принцип контроля доступа" к данным, основанный на присвоении субъектам и объектам доступа классификационных меток (далее — метки), определяющих уровни доступа. В основе данного требования лежит мандатная модель многоуровневого разграничения доступа Белла-Ла Падулы [3] (далее — мандатная модель).

Действующие на настоящее время нормативные документы, определяющие методы и средства защиты от несанкционированного доступа к информации, разрабатывались достаточно давно. Они были ориентированы на технологию централизованной обработки информации на мэйнфреймах с предоставлением оператору текстового интерфейса. Применение требований этих документов к современным ОС общего назначения приводит к ряду трудностей не только технического, но и концептуального характера.

Наиболее наглядным примером является графическая система, которая не только является технически сложным программным слоем, но и меняет принципы работы пользователя с информацией. Поэтому при разработке ОС с графическим интерфейсом, в которой используется мандатная модель, **актуальной является задача программной реализации мандатной модели многоуровневого разграничения доступа в графической системе.** Ее решение должно обеспечить выполнение положений мандатной модели при обработке конфиденциальной информации с предоставлением пользователю эффективного графического интерфейса.

В случае использования для отмеченных выше целей ОС без графического интерфейса, все потоки информации сводятся к обращениям процессов к файлам и к взаимодействию процессов друг с другом с помощью традиционных средств IPC (Inter-Process Communication) и более современных сетевых механизмов. Графический интерфейс, напротив, привносит совершенно новый механизм передачи информации, который качественно отличается от средств IPC, так как проецируется на физическое устройство — экран монитора. В случае с графическим интерфейсом пользователь работает не с процессом, а с графическими объектами (прежде всего, с окнами), которые порождает данный процесс. В результате появляется новый класс сущностей доступа, на которые должна распространяться используемая мандатная модель.

Наиболее очевидным техническим решением рассматриваемой задачи является создание специализированного графического интерфейса. Такой подход возможен, например, для встраиваемых систем, однако он не применим для ОС общего назначения. Вместе с тем следует принимать во внимание, что именно ОС общего назначения используют на рабочих местах должностные лица, где и востребована задача обработки конфиденциальной информации с помощью графических приложений (включая текстовые документы, таблицы, геоданные и др.).

Трудности решения поставленной задачи усугубляет необходимость обеспечения совместимости с существующими общепринятыми приложениями, таким как текстовый процессор, электронная таблица, веб-браузер (для работы в том числе и с геоданными). По этой причине разработчик ОС, которая используется для обработки конфиденциальной информации, вынужден уходить от специализированных решений и базироваться на прототипах, которые стали стандартом де-факто. В случае разработки такой ОС на основе UNIX или Linux стандартной графической системой является X Window System (далее — X Window).

Вопросы реализации мандатной модели в системе X Window

Система X Window включает в себя X-сервер и X-клиентов, которые общаются друг с другом с помощью X-протокола. X-сервер поддерживает устройства ввода—вывода (экран, клавиатура, мышь и др.), а X-клиенты представляют собой графические приложения, которые обращаются к X-серверу, чтобы получить данные от пользователя или вывести информацию на экран.

С точки зрения подлежащей решению задачи важны следующие обстоятельства: система X Window является многооконной; она построена на основе сетевой архитектуры клиент—сервер; ее главный компонент (X-сервер) выполняется в пространстве пользователя. Графические окна при этом, в зависимости от конкретной операции, могут выступать в качестве как объектов доступа (потому что содержат информацию), так и субъектов (так как соответствуют X-клиентам, выполняющим запросы к X-серверу и другим X-клиентам). Данные особенности полностью определяют круг задач, которые необходимо решить разработчику ОС. К их числу относятся следующие задачи.

Передача меток по сети. Так как система X Window является сетевой, актуальна задача передачи меток по сети. Необходим механизм, который должен прозрачно для приложений передавать метку X-клиента, который пытается соединиться с X-сервером, а на стороне X-сервера нужно обеспечить прием данной метки и ее обработку.

Присвоение меток запускаемым приложениям. Вход пользователя в систему в текстовом режиме приводит к созданию сессии, прочно связанной с псевдотерминалом. Пользователь может указать желаемый уровень в рамках своего уровня доступа, после чего данный уровень не меняется на протяжении всей сессии: все процессы, запущенные пользователем, а также псевдотерминал работают с выбранным уровнем. В случае необходимости организовать работу с несколькими уровнями доступа пользователь либо запускает параллельно второй псевдотерминал, независимый от первого, либо выходит из системы и осуществляет вход повторно.

Модель поведения пользователя в графическом режиме принципиально другая, так как появляются: управление мышью (клик, перетаскивание); понятие окна и работы одновременно с несколькими окнами; буфер обмена; понятие фокуса ввода для клавиатуры. В результате вход в систему и старт графической сессии становится сложным процессом и включает в себя: запуск экземпляра X-сервера; запуск сессионных пользовательских служб; запуск графической оболочки и приложений (панель, менеджер окон, рабочий стол).

Для того чтобы сохранить общепринятую парадигму графического интерфейса (перетаскивание мышью, буфер обмена, многооконность), необходимо обеспечить одновременный запуск графических приложений с различными уровнями доступа в рамках текущей сессии.

Авторам импонирует подход, при котором пользователь выполняет вход в графическую сессию с минимальным уровнем доступа, а затем получает возможность запускать приложения с любым позволенным ему уровнем. Трудность заключается в том, что графическая сессия, обеспечивающая взаимодействие окон, была запущена с минимальным уровнем. Однако в то же время она должна поддерживать взаимодействие окон с различными уровнями.

Визуализация меток. Так как система X Window является многооконной, существует специальный X-клиент, который занимается управлением окон и именуется менеджером окон. Менеджер окон осуществляет отображение рамки окна, обеспечивает возможность перемещения, переключения и сворачивания окон, в то время как содержимое окна отобра-

жается приложением. При реализации мандатной модели менеджер окон должен визуально оповещать пользователя о метке окна, с которым тот работает.

Совместно используемые каталоги. Исторически сложилось так, что графические приложения в фоновом режиме создают большое число служебных файлов как во временных каталогах (например, /tmp), так и в домашних каталогах пользователей. При реализации мандатной модели возникают два вопроса: изменение уровня доступа во времени и одновременная работа с несколькими уровнями.

Если приложение в разные моменты времени запускается на разных уровнях доступа, то оно не сможет получать полный доступ на чтение и запись к собственным конфигурационным файлам.

Если приложение является многооконным, то при запуске оно пытается определить, был ли запущен экземпляр данного приложения ранее. Для этого используют временные файлы. Чтобы одновременно работать с таким приложением на разных уровнях, необходимо запускать его несколько независимых экземпляров.

Отмеченные выше вопросы можно разрешить путем адаптации приложений для работы в многоуровневой среде. Однако такой подход существенно снижает совместимость с прототипами и повышает трудозатраты при формировании рабочего пространства пользователя.

Служебные приложения. Менеджер файлов является одним из самых важных компонентов, так как используя его пользователь выполняет поиск нужных документов и фактически "общается" с компьютером.

Когда в файловой системе появляются файлы с различными уровнями доступа, это приводит менеджера файлов к целому ряду сложностей. К их числу относятся перечисленные далее.

- Обычно менеджер файлов представляет собой один многопоточный процесс с несколькими окнами или вкладками. Это обстоятельство затрудняет работу с файлами с различными метками, так как нужно запускать сразу несколько менеджеров файлов с соответствующими уровнями доступа.
- Необходимо, аналогично менеджеру окон, осуществлять визуализацию меток файлов для повышения удобства работы пользователя.
- Открытие файла двойным щелчком мыши представляет собой сложную операцию, включающую сканирование файла для определения его типа, запрос в системную базу данных типов файлов и запуск подходящего приложения. Менеджер файлов может определить, каким приложением можно открыть файл, но не может определить, с каким уровнем нужно запустить приложение.
- При копировании, удалении и перемещении файлов в корзину метка каждого файла должна сохраняться.

Существующие подходы к решению задачи

Рассматриваемая задача неоднократно решалась как зарубежными, так и отечественными разработчиками в разных ОС класса UNIX. Однако сочетание сложности задачи и гибкости UNIX привело к появлению совершенно не похожих друг на друга подходов к реализации. Приведем несколько характерных примеров.

В ОС Solaris 10 с расширениями безопасности в основе построения мандатной многоуровневой системы защиты лежала концепция изолированных зон, которым присваивались уникальные метки. Пользователь при входе в систему выбирал желаемую метку и графическая система создавала для этого пользователя сессию в зоне с соответствующей меткой [4].

Среди отечественных ОС первой разработкой в данной области является МСВС 3.0 (на базе ядра Linux ветви 2.4) [5]. В данной ОС на уровне X-сервера была реализована система изолированных пространств, каждое из которых соответствовало определенному уровню безопасности. Для всех X-клиентов (включая их свойства, события и другие сущности X-протокола) обмен информацией между которыми был разрешен, создавалась иллюзия того, что в графической системе существуют только они [6].

В ядрах Linux, начиная с ветви 2.6, появился механизм модулей безопасности LSM (Linux Security Modules), который позволил разделить процесс управления доступом субъектов к объектам на две части: принятие решения о доступе и исполнение этого решения [7]. С программной точки зрения LSM представляет собой набор функций обратного вызова в пространстве ядра и интерфейс для подключения модулей безопасности. Теоретически к ядру возможно подключить несколько модулей одновременно, каждый из которых будет реализовывать собственную модель управления доступом. В качестве одного из модулей широко применяется система мандатного управления доступом SELinux.

По аналогии с механизмом LSM, для X-протокола было разработано расширение X Access Control Extension (XACE), которое является набором функций обратного вызова в пространстве X-сервера и предоставляет интерфейс для взаимодействия с другими расширениями, позволяющими управлять доступом X-клиентов на уровне X-сервера.

В случае использования SELinux на уровне ядра, на уровне X-сервера через расширение XACE подключается модуль XSELinux [8]. Тогда, с точки зрения SELinux, X-сервер становится менеджером графических объектов (окон, свойств, событий, экрана, указателя, курсора, клавиатуры и подобных им), аналогично тому, как файловая подсистема ядра является менеджером файловых объектов. При этом X-сервер является менеджером в пространстве пользователя. В результате, решения о доступе при взаимодействии X-клиентов принимаются системой SELinux на уровне ядра, а исполняются X-сервером в пространстве пользователя.

Представленные выше примеры иллюстрируют принципиально разные подходы к управлению доступом X-клиентов в ОС с многоуровневой моделью безопасности.

В первом случае используется механизм виртуализации на уровне ОС, а задача организации многоуровневого доступа субъектов к объектам сводится к построению изолированных пространств имен. Строго говоря, требования нормативных документов [2] не выполняются.

Во втором случае используется некое подобие виртуализации на уровне X-сервера. Фактически в ОС часть решений о доступе принимается ядром, а часть — X-сервером, который становится доверенным приложением.

В третьем случае применен гибкий подход по разделению процессов принятия и исполнения решений о доступе. Введены дополнительные классы объектов, отражающие сущности X-протокола, и определены соответствующие им права доступа. Экземплярам классов присваиваются контексты безопасности — аналоги классификационных меток, которые анализируются при принятии решений о доступе.

Рассмотренные примеры не отражают все многообразие возможных вариантов решения поставленной задачи, но подтверждают ее актуальность и необходимость проведения дальнейших исследований.

Новые аспекты рассматриваемой задачи

С развитием сетевых возможностей ОС значительную часть времени пользователи проводят внутри веб-браузера, как основного средства доступа к веб-приложениям. Большинство геоинформационных систем на текущий момент времени используют веб-браузер для отображения и редактирования данных. Многие повседневные задачи, например, редактирование документов, просмотр и отправка электронной почты — решают при помощи веб-приложений.

Фактически, современный веб-браузер выполняет ряд функций ОС. Во-первых, он предоставляет среду выполнения для веб-приложений, скрывая от них детали реализации ОС, точно также как ОС является средой выполнения, скрывающей особенности аппаратной платформы. Во-вторых, он управляет доступом веб-приложений друг к другу и к ресурсам ОС, также как ОС управляет доступом процессов друг к другу, а также к файлам и аппаратным ресурсам. Поэтому веб-браузер должен обладать собственными механизмами защиты, которые должны быть интегрированы с механизмами защиты ОС [9].

Таким образом, задача реализации мандатной модели в графической системе выходит на новый концептуальный уровень. К числу новых задач, связанных с работой веб-приложений, относятся:

- поддержка мандатной модели, реализованной в ОС: веб-браузер должен корректно функционировать в условиях применения принятой в системе модели разграничения, а также обеспечить соблюдение данной модели веб-приложениями, выполняющимися внутри веб-браузера;
- возможность взаимодействия веб-приложений и средств защиты ОС;
- контроль взаимодействия веб-приложений, выполняющихся в разных вкладках веб-браузера, в соответствии с принятой в системе моделью разграничения;
- изоляция веб-приложений друг от друга;
- поддержка возможности передачи метки по сети при использовании системных механизмов назначения метки окну и сокету;
- визуализация меток вкладок веб-браузера.

Выводы

В статье сформулирована задача реализации мандатной модели многоуровневого разграничения доступа в графической системе и определены первоочередные шаги, направленные на ее решение в ОС Linux с использованием системы X Window.

Очевидно, что дальнейшее развитие систем обработки многоуровневой информации будет связано с переходом от графических приложений, которые установлены локально и исполняются в среде ОС, к веб-приложениям, которые загружают с удаленного сервера и исполняют в среде веб-браузера. По этой причине в веб-браузере также необходимо реализовать механизмы поддержки мандатной модели многоуровневого разграничения доступа.

Список литературы

1. **Федеральный** закон РФ от 21 июля 1993 г. № 5485-ФЗ "О государственной тайне".

2. **Гостехкомиссия** России. Руководящий документ. Средства вычислительной техники. Защита от несанкционированного доступа к информации. Показатели защищенности от несанкционированного доступа к информации. М.: Военное издательство, 1992.

3. **Баранов А. П., Борисенко Н. П., Зегжда П. Д., Ростовцев А. Г., Корт С. С.** Математические основы информационной безопасности. М.: Издательство Агентства "Яхтсмен", 1997.

4. **Faden G.** Solaris Trusted Extensions: Architectural Overview. Sun Microsystems white paper, 2006. URL: http://opensolaris.org/os/community/security/projects/tx/.

5. **Тюлин А. Е., Жуков И. Ю., Ефанов Д. В.** На страже конфиденциальной информации // Открытые системы. 2001. № 10. URL: http://www.osp.ru/os/2001/10/180520/

6. **Жуков И. Ю., Ефанов Д. В.** Реализация принципа мандатного управления доступом в графической среде. Методы и технические средства обеспечения безопасности информации // Материалы XV Общероссийской научно-технической конференции. СПб.: Изд-во Политехнического университета, 2006. С. 118—119.

7. **Ефанов Д. В.** Модули безопасности Linux, как стандартный метод реализации политик безопасности // Проблемы информационной безопасности. Компьютерные системы. 2007. № 2. С. 118—122.

8. **Walsh E.** Application of the Flask Architecture to the X Window System Server // Proc. of SELinux Symposium 2007. URL: http://www.nsa.gov/research/_files/selinux/papers/xorg07-abs.shtml.

9. **Ефанов Д. В., Шустова Л. И., Рощин П. Г., Алексеенков А. Г.** Метод обработки многоуровневой информации веб-приложениями в браузере защищенной ОС "Заря" // Информационные и телекоммуникационные технологии. 2013. № 19. С. 43—49.

УДК 004.056

Ю. К. Язов, д-р техн. наук, проф., **А. Л. Сердечный**, ст. науч. сотр., ГНИИИ ПТЗИ ФСТЭК России, г. Воронеж,
e-mail: gniii@fstec.ru

К разработке методов количественного оценивания эффективности ложных информационных систем

Предложены показатели и аналитические соотношения для количественного оценивания эффективности использования ложных информационных систем, построенных с использованием средств виртуализации, как средств защиты реальных систем от сетевых атак. В качестве основного показателя эффективности используется относительное снижение вероятности несанкционированного доступа к ресурсам защищаемой информационной системы за счет предоставления потенциальному нарушителю использования ложной информационной системы. В процессе использования при этом обеспечивается допустимая загруженность вычислительных ресурсов разрабатываемой системы.

Ключевые слова: *ложная информационная система, средство виртуализации, несанкционированный доступ, эффективность защиты, показатель оценки, вычислительный ресурс*

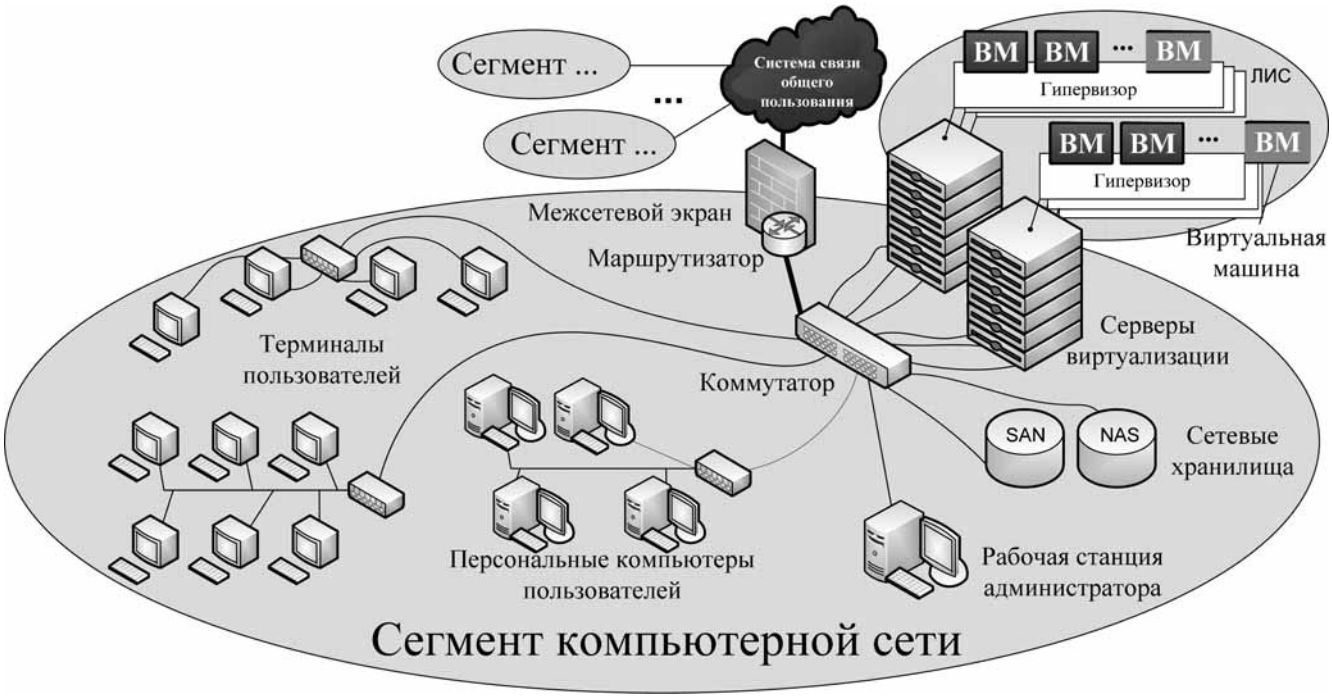
Y. K. Yazov, A. L. Serdechnyy

By the Development of Quantitative Methods of Effectiveness Honeynets Evaluation

The article proposes indicators and analytical relations for the quantitative evaluation of the effectiveness of the use of honeynets built with usage of virtualization as a means to protect production systems from network attacks. The main performance indicator used by the relative decline the possibility of unauthorized access to protected resources information system by providing a potential attacker using honeynets. During usage it ensures permissible load of computational resources of the developed system.

Keywords: *honeynets, virtualization, unauthorized access, efficiency defense, indicator to assess, computational resource*

В условиях стремительной эволюции способов и средств реализации сетевых атак, в том числе с использованием вредоносного программного обеспечения [1], активное развитие получило направление разработки средств защиты информации, основанных на стратегии "обмана" — предоставлении потенциальному нарушителю виртуальных прототипов (аналогов) используемых для этого систем. Такие средства в статье названы ложными информационными системами (ЛИС). Толчком в развитии ЛИС послужило появление на IT-рынке адекватных средств виртуализации. Под средствами виртуализации подразумевается комплекс программных продуктов (например, VMWare vSphere, Citrix XenServer, VirtualBox, VMware Workstation и др.), позволяющих создать виртуальную инфраструктуру и управлять ею. Первоначально средства виртуализации предназначались в основном для снижения ресурсозатрат на разработку, установку и эксплуатацию аппаратного обеспечения информационных систем, а также были ориентированы на повышение их отказоустойчивости и производительности. В настоящее время все больше внимания уделяется применению средств виртуализации в интересах защиты информации в информационных системах, в том числе на основе создания ЛИС (рисунок). В частности, в соответствии с нормативным правовым актом "Требования о защите информации, не составляющей государственную тайну, содержащейся в государственных информационных системах" [2], утвержденным приказом № 17 ФСТЭК России от 11.02.13 г. применение ЛИС регламентируется в ин-



ЛИС в составе компьютерной сети

тересах защиты информации в государственных информационных системах.

Вместе с тем при построении ЛИС возникает целый ряд вопросов методологического характера, связанных, например:

- с обеспечением высокой степени подобия ЛИС реальной информационной системе, а также конфиденциальности защищаемой информации, которая обрабатывается в этой информационной системе;
- с соблюдением баланса между обеспечением конфиденциальности и доступности защищаемой информации, при этом нарушение доступности рассматривается как фактор, обусловленный потреблением вычислительных ресурсов защищаемой информационной системы в интересах обеспечения функционирования ЛИС.

Решение перечисленных вопросов непосредственно связано с оценкой эффективности ЛИС. Поскольку ЛИС применяется в целях отвлечения нарушителя на ложный информационный ресурс, создаваемый с использованием средств виртуализации, то под эффективностью ЛИС понимается степень достижения указанной цели при условии, что ЛИС не влияет существенным образом на функционирование защищаемой информационной системы.

До настоящего времени методический подход к оцениванию эффективности ЛИС, построенных с использованием средств виртуализации, не разрабатывался. В данной статье предлагается возможный подход к такому оцениванию, основанный на вероятностной оценке возможности предотвращения несанкционированного доступа (НСД) к защищаемой информации за счет использования ЛИС при выполнении условия отсутствия превышения затрат вычислительных ресурсов информационной системы (ИС) установленного (недопустимого) уровня ($R \leqslant R_{\text{lim}}$).

Эффективность ЛИС как средства защиты по аналогии с расчетами, приведенными в работе [3], может быть рассчитана с использованием разностного показателя:

$$\eta_{\Delta}(t) = \begin{cases} P_D^{(2)}(t) - P_D^{(1)}(t), & \text{если } R \leqslant R_{\text{lim}}; \\ 0, & \text{если } R > R_{\text{lim}}, \end{cases} \qquad (1)$$

или относительно-разностного показателя

$$\eta(t) = \begin{cases} \dfrac{P_D^{(2)}(t) - P_D^{(1)}(t)}{P_D^{(2)}(t)} = 1 - \dfrac{P_D^{(1)}(t)}{P_D^{(2)}(t)}, \\ \text{если } R \leqslant R_{\text{lim}} \text{ и } P_D^{(1)}(t) \leqslant P_D^{(2)}(t); \\ 0, \text{ если } R > R_{\text{lim}} \text{ или } P_D^{(1)}(t) > P_D^{(2)}(t), \end{cases} \qquad (2)$$

где $P_D^{(2)}(t)$ — вероятность того, что в условиях отсутствия ЛИС за время t будет реализована угроза НСД в ИС;

$P_D^{(1)}(t)$ — вероятность того, что в условиях функционирования ЛИС за время t будет реализована угроза НСД в ИС;

$R \leqslant R_{\text{lim}}$ — условие, определяющее то, что в течение периода времени t на функционирование ЛИС затрачены вычислительные ресурсы R информационной системы в пределах допустимого уровня R_{lim}.

Вероятность $P_D^{(1)}(t)$ в общем случае зависит от вероятностей внедрения средств НСД в ИС; выявления объекта в составе ИС и принятия его в качестве объекта атаки (выявления целевого объекта); преодоления средств защиты целевого объекта; получения доступа к защищаемой информации, однако в интересах исследования ЛИС полагается, что в условиях отсутствия ЛИС у нарушителя всегда имеется возможность внедрения вредоносной программы, преодоления средств защиты целевого объекта и получения доступа к защищаемой информации, при этом вероятность $P_D^{(1)}(t)$ определяется только вероятностью выявления целевого объекта.

Следует отметить, что если вероятность НСД в условиях отсутствия ЛИС близка к единице, то значения показателей эффективности совпадают и определяются только значениями вероятностей $P_D^{(1)}(t)$. Таким образом, если положить, что при отсутствии ЛИС вероятность НСД к целевому объекту равна 1, то с учетом (1) и (2) получим:

$$\eta_{\Delta}(t) = \eta(t) = \begin{cases} 1 - P_D^{(1)}(t), & \text{если } R \leqslant R_{\text{lim}}; \\ 0, & \text{если } R > R_{\text{lim}}, \end{cases} \qquad (3)$$

Рассмотрим подход к оценке вероятности $P_D^{(1)}(t)$ НСД к защищаемой информации, обрабатываемой только одним компьютером (такой компьютер назовем целевым объектом) в составе ИС, в условиях применения ЛИС. При реализации НСД нарушитель или вредоносная программа осуществляет случайный поиск компьютера, который является целью атаки и распознает найденный им объект как целевой. При этом поиск может начинаться с любого элемента ИС, или с ложного элемента в составе ЛИС, формируемого виртуальной машиной. Пусть в составе ИС есть N_{Tr} истинных объектов, ЛИС формирует N_F ложных объектов, а на анализ k-го объекта на предмет отнесения его к целевому тратится время τ_{ak}. Возможное число сочетаний объектов, которые могут анализироваться до того, как нарушитель попадет на целевой объект или на ложный объект, сходный с целевым, рассчитывается по формуле

$$J = \sum_{i=1}^{N_{Tr}+N_F-1} C^i_{N_{Tr}+N_F-1},$$

где $C^i_{N_{Tr}+N_F-1}$ — число сочетаний из i по $N_{Tr}+N_F-1$.

Кроме того, необходимо отметить, что в сочетаниях участвуют только те объекты, которые могут быть обнаружены нарушителем. Так, если нарушитель для обнаружения объектов использует ping-запросы, то сочетания определяются только из объектов, которые имеют возможность ответить на такие запросы.

Вероятность того, что доступ будет осуществлен хотя бы по одному из J сочетаний, определяется из соотношения

$$P_D^{(1)}(t) = \frac{1}{J+1} \sum_{1}^{J} P_j^{(\text{НСД})}(t)\, \bar{P}_{\text{расп}}(k/M_j),$$

где $P_j^{(\text{НСД})}(t)$ — вероятность того, что за время t нарушитель или вредоносная программа успеют проанализировать все объекты в j-м сочетании и получат доступ к целевому k-му объекту; $\bar{P}_{\text{расп}}(k/M_j)$ — вероятность того, что никакой нецелевой объект из j-го сочетания не будет распознан как целевой k-й; M_j — множество объектов в j-м сочетании.

Далее следует, что

$$\bar{P}_{\text{расп}}(k/M_j) = \begin{cases} \prod\limits_{\substack{n \in m \\ n \neq k}} [1 - P_{\text{расп}}(k/n) & M_j \neq \{k\}]; \\ 1 & M_j = \{k\}, \end{cases}$$

где $P_{\text{расп}}(k/n)$ — вероятность того, что нецелевой объект из j-го сочетания будет распознан как целевой k-й объект (условная вероятность перепутывания n-го объекта с k-м).

При экспоненциальном приближении для плотности распределения вероятности случайного времени анализа вероятность НСД к целевому объекту по j-му сочетанию может быть оценена по формуле

$$P_j^{(\text{НСД})}(t) = 1 - \exp\left\{-\frac{1}{\chi'_j(s)\big|_{s=0}}\right\},$$

где $\chi'_j(s)$ — производная от характеристической функции суммарного времени анализа объектов, входящих в j-е сочетание,

$$\chi_j(s) = \prod_{n \in j} \frac{1}{(1 + s\bar{\tau}_{an})}.$$

При этом

$$\chi'_j(s)\big|_{s=0} = \sum_{n=1}^{N_j} \bar{\tau}_{an},$$

где N_j — число объектов j-м сочетании; $\bar{\tau}_{an}$ — среднее время анализа k-го объекта на предмет его отнесения к целевому.

Для оценки вероятностей правильного распознавания целевого и нецелевых объектов (отнесения нецелевого объекта к множеству нецелевых) может быть использована теория статистического распознавания образов [4]. Она основана на выявлении признаков распознавания и принятия по их совокупности решения о принадлежности анализируемого объекта к целевому. При этом одни признаки являются детерминированными, а другие могут быть представлены в виде непрерывных случайных величин. Значения этих величин лежат в некотором интервале, выход за пределы которого фиксируется как аномальный факт, не соответствующий нормальному функционированию ИС.

Характерно, что признаки распознавания могут быть одинаковыми для разных рабочих станций и принадлежать как истинным объектам, так и объектам ложным, которые имитируются с использованием ЛИС. Наличие неопределенности в принадлежности признаков к конкретному объекту приводит к необходимости использования статистической процедуры распознавания. В основе этой процедуры лежит критерий максимума апостериорной вероятности. Согласно этому критерию по набору признаков H принимается решение в пользу k-го объекта, если выполняется условие

$$P(k/H) > P(n/H),\ n = \overline{1, N_{Tr} + N_F},\ n \neq k, \qquad (4)$$

в соответствии с которым апостериорная вероятность отнесения этой совокупности признаков для k-го объекта больше, чем для других объектов.

Апостериорная вероятность рассчитывается по формуле Байеса [4]

$$P(k/H) = \frac{P(k)P(H/k)}{\sum\limits_{n=1}^{N_{Tr}+N_F} P(n)P(H/n)},$$

где $P(n)$ — априорная вероятность появления совокупности признаков для n-го объекта.

Тогда вероятность распознавания объекта как целевого (k-го) по совокупности признаков H определяется следующим образом:

$$P_{\text{расп}}(k/n) = \sum_{n=1}^{N_{Tr}+N_F} \delta_{nk} P(H/n),$$

где δ_{nk} — символ Кронекера, равный 1 при $n = k$, когда выполняется условие (4), и 0 при $n \neq k$.

Если целевой объект один, то априорная вероятность для него определяется следующим образом:

$$P(k) = \frac{1}{N_{Tr} + N_F},$$

а априорные вероятности для каждого из остальных объектов по формуле

$$P(n) = \frac{N_{Tr}^{(n)} + N_F^{(n)}}{N_{Tr} + N_F},$$

где $N_{Tr}^{(n)}$ и $N_F^{(n)}$ — число объектов, функционально схожих с n-м объектом в составе ИС, и имитируемых с помощью ЛИС соответственно.

Если при отнесении объекта к целевому набору используется совокупность признаков H, то вероятность того, что для n-го объекта такие признаки имеют место, определяется из соотношения

$$P(H/n) = \prod_{r \in H} P_r,$$

где P_r — вероятность того, что n-му объекту присущи все признаки из совокупности H.

Если наличие признака является детерминированным событием, то вероятность P_r принимает значение 1, если признак имеет место, и равна 0, если признак отсутствует. При нулевом значении вероятности P_r рассматриваемый объект из процедуры статистического распознавания выпадает. Если значение признака является случайным, распределенным в некотором интервале, то вероятность определяется по плотности распределения вероятности по аналогии с методологией, изложенной в работе [5]. При отсутствии данных о плотности распределения используется равномерное распределение значений признака в заданном интервале.

Пример используемых признаков распознавания для рабочей станции в составе ИС приведен в табл. 1.

Наконец, для оценки эффективности ЛИС с использованием предложенных показателей (см. (3)) необходимо оценить условие $R \leqslant R_{\text{lim}}$ того, что в течение времени t на функционирование ЛИС затрачиваются вычислительные ресурсы ИС в пределах допустимого уровня R_{lim}. К вычислительным ресурсам в данном случае относятся [5]: объем свободной оперативной памяти, необходимой для работы истинных виртуальных машин для каждого сервера виртуализации; резерв свободного процессорного времени, необходимый для надежной работы каждого сервера виртуализации; доля зарезервированного объема трафика, который может передаваться в ИС. Ни один из указанных ресурсов не должен превысить допустимый уровень. С учетом изложенного вычислительный ресурс R представляет собой векторный показатель.

Таблица 1

Примеры признаков, используемых для распознавания нарушителем рабочих станий информационной системы как целевых объектов атаки

№ пп	Признак	Значение вероятности принадлежности признака P_r	Характер признака
1	Соответствие операционной системы рабочей станции семейству операционных систем, к которому принадлежат операционные системы других хостов в составе защищаемой ИС	$P_{r1} = 1$, если принадлежит семейству; $P_{r1} = 0$ в противном случае	Детерминированный
2	Наличие открытых сетевых портов 135 и 445 протокола TCP	$P_{r2} = 1$, если порт открыт; $P_{r2} = 0$, если порт закрыт	
3	Наличие отправки сетевых пакетов в рамках сеанса связи, устанавливаемого по протоколу TCP	$P_{r3} = 1$, если имеет место сеанс связи по протоколу TCP; $P_{r3} = 0$ в противном случае	
4	Нахождение в заданном пределе разницы интенсивности ARP-запросов узла и нормы, характеризующей интенсивность ARP-запросов рабочей станции реального пользователя	$P_{r4} = \int_{\bar{\lambda}_{ARP} - \frac{\Delta\lambda_{ARP}}{2}}^{\bar{\lambda}_{ARP} + \frac{\Delta\lambda_{ARP}}{2}} w\left(\lambda_{ARP}\right) d\lambda_{ARP}$	Случайный, распределенный в диапазоне $\Delta\lambda_{ARP}$; $\bar{\lambda}_{ARP}$ — средняя интенсивность ARP-запросов
5	Нахождение в заданном пределе разницы интенсивности NetBIOS-запросов узла и нормы, характеризующей интенсивность NetBIOS-запросов рабочей станции реального пользователя	$P_{r5} = \int_{\bar{\lambda}_{NetB} - \frac{\Delta\lambda_{NetB}}{2}}^{\bar{\lambda}_{NetB} + \frac{\Delta\lambda_{NetB}}{2}} w\left(\lambda_{NetB}\right) d\lambda_{NetB}$	Случайный, распределенный в диапазоне $\Delta\lambda_{NetB}$; $\bar{\lambda}_{NetB}$ — средняя интенсивность NetBIOS-запросов
6	Несоответствие аппаратного адреса узла диапазону аппаратных адресов, зарезервированных за производителями средств виртуализации	$P_{r6} = 1$, если не соответствует; $P_{r6} = 0$ в противном случае	Детерминированный
7	Отсутствие сигнатур поведения известных "программ-ловушек"	$P_{r7} = 1$, если отсутствует; $P_{r7} = 0$ в противном случае	
8	Проявление сетевой активности узла в нехарактерные для защищаемой ИС периоды времени	$P_{r8} = 1$, если имеет место сетевая активность; $P_{r8} = 0$ в противном случае	

Примечание. w — плотность распределения случайной величины, характеризующей значение соответствующего признака.

Условие $R \leqslant R_{\text{lim}}$ может быть определено при помощи следующего правила:

$$R \leqslant R_{\text{lim}},$$

$$\text{если } \left(V_s^{(real)} \leqslant V_s^{(\min)}\right) \cap \left(T_s^{(real)} \leqslant T_s\right) \cap \left(Q_s^{(real)} \leqslant Q_s\right);$$

$$R > R_{\text{lim}},$$

$$\text{если } \left(V_s^{(real)} > V_s^{(\min)}\right) \cup \left(T_s^{(real)} > T_s\right) \cup \left(Q_s^{(real)} > Q_s\right),$$

где $V_s^{(real)}$ — объем свободной оперативной памяти s-го сервера виртуализации; $V_s^{(\min)}$ — минимальный объем свободной оперативной памяти, необходимый для работы s-го сервера виртуализации и запуска истинной виртуальной машины; $T_s^{(real)}$ — относительный резерв свободного процессорного времени s-го сервера виртуализации; T_s — относительный резерв свободного процессорного времени s-го сервера виртуализации, необходимый для обеспечения функционирования защищаемой ИС (как правило, не превышает 25 % процессорного времени s-го сервера виртуализации [5]); $Q_s^{(real)}$ — относительный резерв объема трафика s-го сервера виртуализации, который остается в результате развертывания и функционирования гипервизора, истинных и ложных виртуальных машин; Q_s — доля объема зарезервированного трафика s-го сервера виртуализации, при котором отсутствуют сбои в работе системы вследствие перегрузки каналов связи в ИС (как правило, не превы-

Таблица 2

Результаты расчета вспомогательных параметров, используемых для нахождения эффективности ЛИС

Номер сочетания объектов j	Множество объектов, входящих в j-е сочетание M_j	Совокупность признаков H	Вероятность успешного анализа за время t всех объектов в j-м сочетании $P_j^{(\text{НДС})}(t)$	Вероятность правильного распознавания целевого объекта $P_{\text{расп}}(\text{У1}/\text{У1})$	Вероятность того, что никакой нецелевой объект из j-го сочетания не будет распознан как целевой $\bar{P}_{\text{расп}}(\text{У1}/M_j)$
1	{У1}	H_1	≈1 (при t > 30 с)	$P_{r1}^{\text{У1}} P_{r2}^{\text{У1}} = 1 \cdot 1 = 1$	1
		H_2		$P_{r1}^{\text{У1}} P_{r2}^{\text{У1}} P_{r6}^{\text{У1}} = 1 \cdot 1 \cdot 1 = 1$	
2	{У1, У2}	H_1	≈1 (при t > 1 мин)	1	$1 - P_{r1}^{\text{У2}} P_{r2}^{\text{У2}} = 1 - 0 \cdot 0 = 1$
		H_2			$1 - P_{r1}^{\text{У2}} P_{r2}^{\text{У2}} P_{r6}^{\text{У2}} = 1 - 0 \cdot 0 \cdot 0 = 1$
3	{У1, У3}	H_1	≈1 (при t > 1 мин)		$1 - P_{r1}^{\text{У3}} P_{r2}^{\text{У3}} = 1 - 1 \cdot 1 = 0$
		H_2			$1 - P_{r1}^{\text{У3}} P_{r2}^{\text{У3}} P_{r6}^{\text{У3}} = 1 - 1 \cdot 1 \cdot 0 = 1$
4	{У1, У4}	H_1	≈1 (при t > 1 мин)		$1 - P_{r1}^{\text{У4}} P_{r2}^{\text{У4}} = 1 - 0 \cdot 0 = 1$
		H_2			$1 - P_{r1}^{\text{У4}} P_{r2}^{\text{У4}} P_{r6}^{\text{У4}} = 1 - 0 \cdot 0 \cdot 0 = 1$
5	{У1, У2, У3}	H_1	≈1 (при t > 2 мин)		$\left(1 - P_{r1}^{\text{У2}} P_{r2}^{\text{У2}}\right)\left(1 - P_{r1}^{\text{У3}} P_{r2}^{\text{У3}}\right) = 0$
		H_2			$\left(1 - P_{r1}^{\text{У2}} P_{r2}^{\text{У2}} P_{r6}^{\text{У2}}\right)\left(1 - P_{r1}^{\text{У3}} P_{r2}^{\text{У3}} P_{r6}^{\text{У3}}\right) = 0$
6	{У1, У2, У4}	H_1	≈1 (при t > 2 мин)		$\left(1 - P_{r1}^{\text{У2}} P_{r2}^{\text{У2}}\right)\left(1 - P_{r1}^{\text{У4}} P_{r2}^{\text{У4}} P_{r6}^{\text{У4}}\right) = 1$
		H_2			$\left(1 - P_{r1}^{\text{У2}} P_{r2}^{\text{У2}} P_{r6}^{\text{У2}}\right)\left(1 - P_{r1}^{\text{У4}} P_{r2}^{\text{У4}} P_{r6}^{\text{У4}}\right) = 0$
7	{У1, У3, У4}	H_1	≈1 (при t > 2 мин)		$\left(1 - P_{r1}^{\text{У3}} P_{r2}^{\text{У3}}\right)\left(1 - P_{r1}^{\text{У4}} P_{r2}^{\text{У4}} P_{r6}^{\text{У4}}\right) = 0$
		H_2			$\left(1 - P_{r1}^{\text{У3}} P_{r2}^{\text{У3}} P_{r6}^{\text{У3}}\right)\left(1 - P_{r1}^{\text{У4}} P_{r2}^{\text{У4}} P_{r6}^{\text{У4}}\right) = 1$
8	{У1, У2, У3, У4}	H_1	≈1 (при t > 2 мин)		$\left(1 - P_{r1}^{\text{У2}} P_{r2}^{\text{У2}}\right)\left(1 - P_{r1}^{\text{У3}} P_{r2}^{\text{У3}}\right) \times$ $\times \left(1 - P_{r1}^{\text{У4}} P_{r2}^{\text{У4}} P_{r6}^{\text{У4}}\right) = 0$
		H_2			$\left(1 - P_{r1}^{\text{У2}} P_{r2}^{\text{У2}} P_{r6}^{\text{У2}}\right)\left(1 - P_{r1}^{\text{У3}} P_{r2}^{\text{У3}} P_{r6}^{\text{У3}}\right) \times$ $\times \left(1 - P_{r1}^{\text{У4}} P_{r2}^{\text{У4}} P_{r6}^{\text{У4}}\right) = 1$

Примечание. Условие затрат вычислительных ресурсов в пределах допустимого уровня $R \leqslant R_{\text{lim}}$ выполняется.

шает 50 % суммарной пропускной способности каналов связи, подходящих к *s*-му серверу виртуализации, однако, для ИС офисного типа данный показатель может быть меньше ввиду малого объема файлов, передаваемых в такой ИС, и отсутствия жестких требований к качеству обслуживания [5]).

Введем следующие обозначения: V_s — общий объем оперативной памяти *s*-го сервера виртуализации; $V_s^{(0)}$, $V_s^{(\text{ВМ})}$, $V_s^{(\text{ГВ})}$ — средний объем свободной оперативной памяти, необходимый для работы истинной, ложной виртуальной машины и гипервизора для *s*-го сервера виртуализации соответственно. Тогда

$$V_s^{(real)} = V_s - V_s^{(\text{ГВ})} - m_F V_s^{(\text{ВМ})} - m_{Tr} V_s^{(0)},$$

где m_{Tr}, m_F — число истинных и ложных виртуальных машин в ИС соответственно, создаваемых с использованием *s*-го сервера виртуализации.

Пусть $T_s^{(\text{ВМ})}$, $T_s^{(0)}$, $T_s^{(\text{ГВ})}$ — относительные затраты процессорного времени на обеспечение функционирования ложной виртуальной машины, истинной виртуальной машины и гипервизора соответственно. Тогда

$$T_s^{(real)} = 1 - T_s^{(\text{ГВ})} - m_F T_s^{(\text{ВМ})} - m_{Tr} T_s^{(0)}. \qquad (5)$$

Обозначим через $Q_s^{(\text{ВМ})}$, $Q_s^{(0)}$, $Q_s^{(\text{ГВ})}$ относительный объем трафика, передаваемый в ИС в результате функционирования, соответственно, ложной виртуальной машины, истинной виртуальной машины и гипервизора.

Тогда, по аналогии с (5):

$$Q_s^{(real)} = 1 - Q_s^{(\text{ГВ})} - m_F Q_s^{(\text{ВМ})} - m_{Tr} Q_s^{(0)}.$$

Рассмотрим пример оценивания эффективности ЛИС для условной компьютерной сети, состоящей из одного целевого объекта, являющегося рабочей станцией (У1), сервера виртуализации (У2), на котором работают две виртуальные машины: ложная виртуальная машина, имитирующая рабочую станцию пользователя (У3), и истинная, не представляющая интереса для злоумышленника (У4). Предположим, что все четыре объекта могут быть выявлены нарушителем, причем порядок выявления таких объектов случайный. Распознавание рабочей станции нарушителем осуществлялось для двух случаев. В первом случае были использованы признаки 1, 2 (см. табл. 1), а во втором — признаки 1, 2, 6 (см. табл. 1). Во втором случае эффективность ЛИС ожидалась равной 0, так как ложная виртуальная машина не обеспечивала должный уровень схожести (подобия) с рабочей станцией по признаку 6 (несоответствие аппаратного адреса узла диапазону аппаратных адресов, зарезервированных за производителями средств виртуализации). Вычислительных ресурсов сервера виртуализации достаточно для обеспечения работы двух виртуальных машин.

В табл. 2 представлены результаты расчета вспомогательных параметров, используемых для нахождения эффективности $\eta(t)$. Для первого случая распознавания эффективность ЛИС $\eta_1(t) = 0{,}5$, а для второго — $\eta_2(t) = 0$.

Таким образом, в статье предложен подход, позволяющий количественно оценивать эффективность ЛИС, использующих средства виртуализации, что позволяет перейти к количественным методам обоснования требований к ним и путей их построения.

Список литературы

1. **Минаков В. А., Сердечный А. Л., Язов Ю. К.** Угрозы безопасности информации и проблемы защиты информации в информационных системах, построенных с использованием облачных технологий // Персональные данные. Электронный журнал. 2012. № 4 (45). Режим доступа: http://www.privacy-journal.ru/article/105/2/1220.

2. **Требования** о защите информации, не составляющей государственную тайну, содержащейся в государственных информационных системах. Утверждены приказом ФСТЭК России от 11.02.2013 № 17 г. Москва.

3. **Язов Ю. К.** Основы методологии количественной оценки эффективности защиты информации в компьютерных сетях. Ростов-на-Дону: СКНЦ ВШ, 2006. 274 с.

4. **Акимов П. С., Бакут П. А., Богданович В. А.** и др.; Теория обнаружения сигналов / Под ред. П. А. Бакута. М.: Радио и связь, 1984. 440 с.

5. **Михеев М. О.** Администрирование VMware vSphere 5. М.: ДМК Пресс, 2012. 504 с.

6. **Левин Б. Р.** Теоретические основы статистической радиотехники. Кн. 1. М.: Сов. радио, 1974. 550 с.

УДК 519.612

К. Ф. Иванова, канд. техн. наук, докторант, Санкт-Петербургский государственный университет,
e-mail: klara.i2010@yandex.ru

Оценка объединенного множества решений задачи на основе интервальной модели Леонтьева

Предлагается подход к решению модельной задачи В. В. Леонтьева, которая описывается интервальной системой линейных алгебраических уравнений на основе применения "знаковой" методики. Получены интервалы для определения валовых компонентов отраслевого хозяйства как результат частного случая решения двух точечных линейных алгебраических систем с положительными квадратными матрицами и с правильными интервалами правой части. Реализуется оценка допустимого решения с соблюдением принципа всеобщности, предшествующего принципу существования, имеющих место при построении внутренней оценки допустимого решения интервальной системой линейных алгебраических уравнений.

Ключевые слова: *интервальная система уравнений, оценка объединенного решения, "знаковая" методика, две точечные линейные системы.*

K. F. Ivanova

Estimation of the United Solution Set of an Interval Leontjev's Model

The approach to the solution of a modelling V. V. Leontjev's problem described by interval system of linear equations (ISLAE) on the basis of application of a "sign" technique is offered. Intervals for total a component of a branch facilities as result of a special case of the solution of two point linear algebraic systems with positive square matrixes and with correct intervals of an interval right-hand side vector are received. The estimation of the united solution set with observance principle of the universality, previous to a principle of the existence, having a place at construction of an internal solution ISLAE is realized.

Keywords: *interval system of the equations, estimation of the tolerable solution set, "sign" technique, two point linear systems.*

Введение

Рассматривается интервальная модель межотраслевого баланса, связанная с эффективностью ведения отраслевого хозяйства. Эту модель, разработанную в 1936 г. В. В. Леонтьевым, называют моделью "затраты—выпуск" [1]. К числу задач данного направления можно отнести многие задачи математической экономики, автоматического управления, технологического проектирования. Предлагаемая математическая модель межотраслевого баланса описывается интервальной системой линейных алгебраических уравнений (ИСЛАУ), коэффициенты матрицы которой и компоненты вектора правой части, представляющего конечный продукт, описываются неточно. Требуется найти решение системы, определяющее валовое распределение произведенной продукции, которое удовлетворяет всем возможным заданным интервалам правой части.

Оценка погрешности решения задачи межотраслевого баланса основана на интервальном представлении входных данных и использовании "знаковой" методики. Интервальный подход к решению подобных задач впервые был предложен Дж. Роном [2]. Теоретическое обоснование и условия разрешимости обобщенных решений детально разработаны С. П. Шарым и И. А. Шарой [3—5]. В работе [3] постановка интервальной задачи заменяется на задачу одной точечной (неинтервальной) системы уравнений в евклидовом пространстве двойной размерности и конструируется специализированный алгоритм — субдифференциальный метод Ньютона, реализующий новый подход

получения решения ИСЛАУ. В работе [4] доказывается, что допустимое решение линейной системы может быть представлено, с одной стороны, как пересечение гиперполос из двухсторонних неравенств и, с другой стороны, как сумма линейных подпространств с ограниченным выпуклым многогранником. В работе [5] показано, что допусковое множество решений является пересечением конечного числа гиперполос, т. е. множеством решений конечной системы двусторонних нестрогих линейных неравенств. Проведенные оценки ИСЛАУ в ряде случае относятся и к системам, определяющим модель межотраслевого баланса.

"Знаковая" методика к оценке погрешности решения системой линейных алгебраических уравнений (СЛАУ) впервые была предложена в работе [6]. В работе просчитывались СЛАУ 2—3 порядков с относительными погрешностями элементов матрицы определенных знаков, сформированных по определенному закону и способных инициировать в ряде случаев значительно большую погрешность решения по сравнению с постоянными относительными погрешностями коэффициентов. В работах [7—10] описаны оценки погрешности интервальных линейных систем с применением "знаковой" методики. В работе [7] рассмотрены оценки погрешности решения ИСЛАУ с неопределенными входными данными по общепринятым интервальным оценкам и "знаковой" методике, в работах [8—9] дан анализ погрешности решений уравнений в частных производных при их численной аппроксимации ИСЛАУ, в работе [10] приведен анализ интервальной задачи линейного программирования.

Новизна данной работы состоит в использовании "знаковой" методики для оценки интервалов решения модельной задачи межотраслевого баланса. Такой подход имеет очевидное преимущество в алгоритмизации и реализации численных расчетов задачи по сравнению с известными оценками моделей, приведенными в работах [3—5, 11]. В соответствии со "знаковой" методикой путем элементарного анализа показано, что формируемые две точечные граничные матрицы исследуемой ИСЛАУ совпадают с интервальными границами исходной интервальной матрицы.

Постановка интервальной задачи "затраты—выпуск" Леонтьева

Модель Леонтьева "затраты—выпуск" предполагает, что объем каждой из n отраслей производства должен удовлетворить всем потребностям в этой отрасли и обеспечить связь с соответствующими уровнями сопряженных отраслей.

Для удобства изложения введены следующие замечания и обозначения:

- полужирным шрифтом размечены интервальные матрицы, векторы и их компоненты;
- прямым шрифтом размечены все интервальные и вещественные матрицы;
- компоненты векторов и матриц обоего вида размечены курсивом;
- R — множество вещественных чисел;
- IR — множество интервальных вещественных чисел;
- $\mathbf{A} \in \mathbf{IR}^{n \times n}$ — интервальная матрица, имеющая n строк, n столбцов; $\mathrm{IR}^{n \times n}$ — интервальное вещественное пространство размерности $n \times n$; $\mathbf{b} \in \mathrm{IR}^n$ — интервальный вектор с n компонентами;
- границы интервальных величин в общем случае отмечены нижним и верхним подчеркиванием и заключены в квадратные скобки;
- границы интервалов векторов и матриц, полученные по "знаковой" методике, снабжены символами (+) и (−).

В точечном (неинтервальном) варианте уравнение межотраслевого баланса имеет вид

$$x = Ax + y,\ x, y \in R^n,\ y \geqslant 0,\ A \in IR^{n \times n}, \qquad (1)$$

где A — продовольственная матрица прямых затрат (технологическая матрица); x — объем произведенной продукции (вектор валового выпуска); y — объем продукции конечного потребления (вектор конечного потребления).

В развернутом виде матрица и векторы имеют следующий вид:

$$x = \begin{pmatrix} x_1 \\ x_2 \\ \dots \\ x_n \end{pmatrix},\ y = \begin{pmatrix} y_1 \\ y_2 \\ \dots \\ y_n \end{pmatrix},\ A = \begin{pmatrix} a_{11}\ a_{12}\ \dots\ a_{1n} \\ a_{21}\ a_{22}\ \dots\ a_{2n} \\ \dots\ \dots\ \dots\ \dots \\ a_{1n}\ a_{n2}\ \dots\ a_{nn} \end{pmatrix}, \qquad (2)$$

где x_i — общий объем продукции i-й отрасли (ее валовой выпуск); a_{ij} — постоянные числа, называемые коэффициентами прямых затрат, и определяющие линейность производства объема продукции x_j, выражаются через произведение элементов матрицы на объем продукции i-й отрасли x_i; $a_{ij}x_i = x_{ij}$, где x_{ij} — объем продукции i-й отрасли, потребляемый j-й отраслью при производстве; y_i — объем продукции i-й отрасли, предназначенный для реализации в непроизводственной сфере — объем потребления.

К основным положениям понятия продуктивности относятся следующие критерии продуктивности.

1. Матрица $A \geqslant 0$ модели межотраслевого баланса является продуктивной, как и сама модель, если для любого вектора $y \geqslant 0$ существует решение $x \geqslant 0$ уравнения (1). Экономически это означает, что любое конечное потребление y можно обеспечить при подходящем валовом выпуске. Перепишем уравнение (1) в виде

$$(E - A)x = y, \qquad (3)$$

где E — единичная матрица. Если обратная матрица $(E - A)^{-1}$ существует, то

$$x = (E - A)^{-1}y. \quad (4)$$

Отсюда следует второй критерий продуктивности.

2. Матрица $A \geqslant 0$ продуктивна только тогда, когда матрица $(E - A)^{-1}$ существует и неотрицательна.

Одним из следствий критериев продуктивности является выполнение условия $\sum_{i=1}^{n} a_{ij} < 1$, которое означает рентабельность j-й отрасли.

Коэффициенты матрицы A могут быть приняты константами только в небольшом временном промежутке. Вектор конечного потребления y также считается известным и постоянным в течение определенного времени. Значения коэффициентов a_{ij}, рассматриваемые за более длительный интервал времени (например, год), уже требуют представления внутри замкнутого интервала, истинный размер которого может приобретать значение из полученного промежутка. Вместе с тем вектор конечного продукта y в течение года также может ежемесячно меняться, но компоненты его остаются в некоторых границах. Вариабельность коэффициентов модели может быть вызвана как ростом, так и падением производства в целом, неточностью технического сбора информации и вычислительными ошибками обработки исходных данных.

В интервальном виде матричная система линейных уравнений является символической записью для совокупностей точечных систем уравнений (2) той же структуры. Коэффициенты при неизвестных и элементы правой части принадлежат соответствующим интервалам, и ИСЛАУ может быть представлена в следующем виде:

$$\mathbf{x} = \mathbf{A}x + \mathbf{y},\ \mathbf{x} \in R^n,\ \mathbf{y} \in \mathbf{IR}^n,\ \mathbf{A} = (\boldsymbol{a}_{ij}) \in \mathbf{R}^{n \times n}. \quad (5)$$

Для интервальной линейной системы уравнений (5) определены различные множества решений [1—3]. Наиболее популярны из них объединенное (внешнее), допустимое (внутреннее) и управляемое. В исследовании, результаты которого представлены в данной статье, рассмотрено объединенное множество решений $\Xi_{\exists\exists}((E - \mathbf{A}), \mathbf{y})$, при котором для всякой матрицы коэффициентов $(E - A) \in (E - \mathbf{A})$ значение произведения $(E - A)x$ не выходит за пределы интервального вектора **y**:

$$\Xi_{\exists\exists}((E - \mathbf{A}), \mathbf{y}) =$$
$$= \{x \in R^n | (\exists \mathbf{A} \in \mathbf{A})(\exists y \in \mathbf{y})((E - \mathbf{A})x = y)\}. \quad (6)$$

Объединенное множество решений системы вида (5), определяется как объединение решений всех точечных систем, входящих в (5).

Моделирование задачи Леонтьева по "знаковой" методике

Как следует из постановки интервальной задачи, должна быть проведена оценка разброса валового продукта x, при котором реализуется попадание решения в интервал изменения вектора конечного продукта y. Таким образом, имеет место оптимальная задача, результатом решения которой является получение максимального разброса компоненты x_i, способной обеспечить заданные пределы изменения компоненты y_i.

Алгоритм на основе "знаковой" методики базируется на определении знаков относительных погрешностей элементов матрицы и компонент правых частей (левой или правой границы), приводящих к максимальному разбросу или возможным погрешностям компонент вектора решения [1, 6]. Этот анализ следует из сочетания знаков элементов матрицы и соответствующих им алгебраических дополнений, которые обеспечивают максимальное линейное приращение детерминанта матрицы исходных измерений СЛАУ или средней матрицы ИСЛАУ.

При постановке задачи считается известным интервал, куда попадают истинные значения $a_{ист}$ и $y_{ист}$ элементов матрицы **A** и компонент вектора правых частей: $\underline{a} \leqslant a_{ист} \leqslant \overline{a}$, $\underline{y} \leqslant y_{ист} \leqslant \overline{y}$. Интервальные коэффициенты $\boldsymbol{a}_{ij} = [\underline{a}_{ij}, \overline{a}_{ij}]$, $\boldsymbol{y}_i = [\underline{y}_i, \overline{y}_i]$ представляются в виде отрезков на вещественной оси с соответствующими границами интервалов неопределенностей. Под положительностью интервальной матрицы понимается условие неотрицательности ее компонент $\mathbf{A} = [\underline{A}, \overline{A}]$. Этот факт означает, что в конечномерном пространстве компоненты матрицы и векторы правых частей задаются своими положительными границами. Раскрывая выражения для отдельных элементов интервальной системы (5), получаем

$$\boldsymbol{a}_{ij} \in [a_{ij} - \varepsilon_{ij}a_{ij},\ a_{ij} + \varepsilon_{ij}a_{ij}],$$
$$\boldsymbol{y}_i \in [y_i - \delta_i y_i,\ y_i + \delta_i y_i],\ \forall i = \overline{1, n},\ \forall j = \overline{1, n}. \quad (7)$$

Для удобства изложения обозначим разность точечных матриц $(E - A)$, участвующих в расчетах, через V, так что интервальное уравнение (5) можно переписать

$$\mathbf{V}x = \mathbf{y}. \quad (8)$$

В дальнейшем будем понимать, что **V** — разность между единичной матрицей E и матрицей, составленной из компонент интервальной матрицы **A**. Основная идея предлагаемой оценки решения x заключается в определении знаков относительных погрешностей ε коэффициентов квадратной матрицы **V** и относительных погрешностей вектора правых частей δ, инициирующих наибольшие отклонения вектора решения x.

Процедура выбора знаков ε и δ построена на анализе квадратной матрицы системы линейных алгебраических уравнений (7) с точечными коэффициентами

$$\mathrm{V}x = y,\ \mathrm{V} \in R^{n \times n},\ y \in R^n. \qquad (9)$$

Полагаем, что точечные матрицы системы (9) невырождены. Граничные коэффициенты системы (9) $v_{ij} = \overline{v}_{ij} + \varepsilon_{ij}\overline{v}_{ij}$ и $y_i = \overline{y}_i + \delta_i\overline{y}_i$, где $\overline{v}_{ij}$, $\overline{y}_{ij}$ — средние значения интервалов. Эта информация позволяет воспользоваться "знаковой" методикой для оценки множества решений системы (7), которая основывается на анализе максимального приращения детерминанта исходной матрицы **V**. Главная, линейная часть приращения определителя матрицы **V** представлена произведением элементов строки (столбца) на соответствующие им алгебраические дополнения. Приращение вычисляется как произведение частных производных определителя матрицы по переменным ε_{ij}, умноженным на приращение $\Delta\varepsilon_{ij}$. Полагаем, что приращение относительной погрешности не превосходит самой погрешности $|\Delta\varepsilon_{ij}| \leqslant \varepsilon_{ij}$, и максимальная поэлементная погрешность не превосходит задаваемого ограничения ε_0: $|\varepsilon_{ij}| \leqslant \varepsilon_0$.

Приращение детерминанта при разложении определителя по *j*-му столбцу в линейном приложении можно записать в виде

$$\Delta\mathrm{detV} = \mathrm{v}_{1j}(1 + \varepsilon_{1j})\mathrm{Vd}_{1j} + \mathrm{v}_{2j}(1 + \varepsilon_{2j})\mathrm{Vd}_{2j} + \ldots + \mathrm{v}_{nj}(1 + \varepsilon_{nj})\mathrm{Vd}_{nj} = \sum_{i=1}^{n} \mathrm{v}_{ij}(1 + \varepsilon_{ij})\mathrm{Vd}_{ij},\ \forall j \subset \{1, 2, \ldots, n\}, \qquad (10)$$

где $\mathrm{V}d_{ij} = (-1)^{i+j}\mathrm{M}_{ij}$ — алгебраические дополнения элемента v_{ij} в detV. Если V — определитель *n*-го порядка, то M_{ij} — минор — определитель $n - 1$ порядка, полученный вычеркиванием *i*-й строки и *j*-го столбца. Отсюда следует, что наибольшее возможное значение (10) в линейном приближении достижимо в том случае, когда относительная погрешность ε_{ij} по знаку совпадает со знаком произведения $\mathrm{v}_{ij}\mathrm{V}d_{ij}$.

Полагаем, что $|\Delta\delta_i| \leqslant \delta_i$ и $|\delta_i| \leqslant \delta_0$ (т. е. заданного значения), тогда максимальное линейное приращение определителя при замене столбца элементов матрицы столбцом правой части можно записать как

$$\Delta\mathrm{detV} = \sum_{i=1}^{n} \mathrm{y}_i\delta_0\mathrm{Vd}_{ij},\ \forall j \subset \{1, 2, \ldots, n\}. \qquad (11)$$

Как и в (10), максимальное линейное приращение (11) в положительном направлении обеспечивается тогда, когда знак δ_i совпадает со знаком произведения $\mathrm{y}_i\mathrm{Vd}_{ij}$. Следствием из выражений для приращений (10) и (11) являются неравенства

$$\begin{cases} \mathrm{sign}(\mathrm{v}_{ij}\varepsilon_{ij}\mathrm{Vd}_{ij}) > 0,\ \text{if } \mathrm{sign}(\varepsilon_{ij}) = \mathrm{sign}(\mathrm{v}_{ij}\mathrm{Vd}_{ij}), \\ \mathrm{sign}(\mathrm{v}_{ij}\varepsilon_{ij}\mathrm{Vd}_{ij}) < 0,\ \text{if } \mathrm{sign}(\varepsilon_{ij}) \neq \mathrm{sign}(\mathrm{v}_{ij}\mathrm{Vd}_{ij}), \\ \mathrm{sign}(\mathrm{y}_i\delta_i\mathrm{Vd}_{ij}) > 0,\ \text{if } \mathrm{sign}(\delta_i) = \mathrm{sign}(\mathrm{y}_i\mathrm{Vd}_{ij}), \\ \mathrm{sign}(\mathrm{y}_i\delta_i\mathrm{Vd}_{ij}) < 0,\ \text{if } \mathrm{sign}(\delta_i) \neq \mathrm{sign}(\mathrm{y}_i\mathrm{Vd}_{ij}). \end{cases} \qquad (12)$$

Положительные значения неравенств (12) определяют наибольшие отклонения решения в положительном направлении, а отрицательные значения — в противоположном. Из выражений (12) возможен формальный переход от интервальных коэффициентов (8) к их точечным значениям выбором только одной границы интервала элемента $\mathbf{V}_{ij}$, в зависимости от знака ε_{ij} или δ_i, на основании которых формируются две новые вещественные матрицы V^- и V^+. В общем случае матрицы V^- и V^+ отличны от граничных матриц $\underline{\mathrm{V}}$ и $\overline{\mathrm{V}}$, когда отрицательные ε_{ij} и δ_i входят в компоненты левых границ элементов матрицы $\underline{\mathrm{V}}$ и вектора $\underline{\mathrm{y}}$, а положительные — в компоненты правых границ матрицы $\underline{\mathrm{V}}$ и $\overline{\mathrm{y}}$. Верификация результатов расчета вектора решения x по "знаковой" методике проводится по формулам Крамера. Для сравнения приводится вычисление компонент вектора x по средним значениям коэффициентов матрицы и компонент правой части:

$$x_j = \frac{\Delta_{\mathrm{y}}}{\Delta},\quad x_j^- = \frac{\Delta_{\underline{\mathrm{y}}}^+}{\Delta^+},\quad x_j^+ = \frac{\Delta_{\overline{\mathrm{y}}}^-}{\Delta^-},\quad j = \overline{1, n}, \qquad (13)$$

где Δ^- и Δ^+ — детерминанты, полученные в сторону убывания и возрастания точечных матриц, обозначенных V^- и V^+, y, y^-, y^+ — среднее значение вектора правой части, левая и правая границы интервала соответственно, подставляемые в детерминанты.

Первое равенство в (13) — вычисление компоненты x_j без возмущения входных данных. Из второго равенства (13) следует, что подстановка вектора правой части, полученного в направлении убывания, в детерминант, полученный в направлении возрастания, приведет к уменьшению компоненты x_j из первого равенства. В третьем равенстве (13) в детерминант, полученный в сторону убывания, подставляется вектор правой части, сформированный в сторону возрастания определителя. Такие простые действия позволяют продемонстрировать, каким образом можно выявить максимальное возрастание относительной погрешности решения $otn1 = \dfrac{\left(x_j^+ - x_j\right)}{x_j}$ и $otn2 = \dfrac{\left(x_j - x_j^-\right)}{x_j}$ в случае неточности задания входных данных.

Возвращаясь к исходной интервальной матрице $\mathbf{A} = [\underline{\mathrm{A}}, \overline{\mathrm{A}}]$, можно отметить, что в соответствии со "знаковой" методикой совпадение границ матриц A^- и $\underline{\mathrm{A}}$, а также A^+ и $\overline{\mathrm{A}}$ может быть только в том случае,

если приращение компонент в сторону увеличения детерминанта матрицы **A** и в направлении его уменьшения совпадает с правыми и левыми границами исходной матрицы. В частности, это оказалось возможным в случае матриц $\underline{A} \geqslant 0$, и $\overline{A} \geqslant 0$, отвечающим критериям продуктивности (условию 1), и положительности обратных точечных матриц $\underline{A}^{(-1)} \geqslant 0$, $\overline{A}^{(-1)} \geqslant 0$ (условию 2).

В соответствии с неравенствами (12) относительные погрешности ε_{ij} и δ_i определенных знаков обусловливают создание двух новых граничных матриц и векторов V^+ и y^+, V^- и y^-, участвующих в формировании допустимого множества решения (6) интервальной системы (8). Границы V^- и V^+ матрицы **V** уже не совпадают с $\underline{V}$ и $\overline{V}$.

Реализация "знаковой" методики по оценке решения системы ИСЛАУ возможна и, тем более точна в том случае, если значения ε_0 и δ_0 значительно меньше 100 % для относительных погрешностей коэффициентов матриц и компонент вектора правой части.

Таким образом, сформирована интервальная система с новыми границами матрицы **V** и вектора **y**

$$\mathbf{V} = [V^-, V^+] \text{ и } \mathbf{y} = [y^-, y^+], \qquad (14)$$

которую можно переписать в виде двух точечных линейных алгебраических систем

$$\begin{cases} V^+ x^- = y^- \\ V^- x^+ = y^+, \end{cases} \qquad (15)$$

где $V^- \in \mathbf{V}$ и $V^+ \in \mathbf{V}$ с векторами $y^- \in \mathbf{y}$ и $y^+ \in \mathbf{y}$.

Покажем, что парное решение системы (15) идентично допустимому оцениванию множества решений (8).

Пример решения задачи Леонтьева с применением "знаковой" методики

Продемонстрируем применение интервального подхода к обработке модели "затраты—выпуск", охватывающей четыре крупных предприятия г. Балашов, выпускающих продукцию и на внешний по отношению к ним рынок [11]: АООТ "Прицеп", МППУ "Водоканал", "Городские электросети", "Городские теплосети". Обрабатывалась информация, собранная за каждый месяц 2000 г. Поскольку потребление продукции теплосетей существенно зависит от времени года, летний и зимний вариант этой модели обрабатывались отдельно. Предлагается для рассмотрения интервальная линейная система с технологической матрицей **A** размера 4 × 4, компоненты которой являются интервалами $\boldsymbol{a_{ij}}$ ($i, j = 1, 2, 3, 4$), и вектором **y** с интервальными компонентами $\boldsymbol{y_i}$ ($i = 1, 2, 3, 4$).

В результате обработки информации о потреблении четырьмя предприятиями собственной продукции и продукции входящих в модель других предприятий были получены две технологические матрицы и векторы конечного потребления для зимнего и летнего периодов.

Зимняя модель:

$$\mathbf{A} = \begin{pmatrix} [0{,}011715;\ 0{,}018671] & [0{,}075467;\ 0{,}076139] & [0{,}024138;\ 0{,}025147] & [0{,}053347;\ 0{,}056022] \\ [0{,}025901;\ 0{,}029653] & [0{,}008862;\ 0{,}009710] & [0{,}012754;\ 0{,}013289] & [0{,}015241;\ 0{,}016007] \\ [0{,}022221;\ 0{,}024818] & [0{,}044478;\ 0{,}051499] & [0{,}008365;\ 0{,}008943] & [0{,}026854;\ 0{,}028202] \\ [0{,}029651;\ 0{,}030991] & [0{,}024861;\ 0{,}025084] & [0{,}014857;\ 0{,}015432] & [0{,}010041;\ 0{,}011160] \end{pmatrix},$$

$$\mathbf{y} = ([703{,}1,\ 723{,}0],\ [405{,}8,\ 410{,}6],\ [637{,}7,\ 669{,}0],\ [472{,}7,\ 498{,}5])^{\mathrm{T}}.$$

На рис. 1 представлен листинг интервального вычисления вектора **x** валовой продукции для зимней модели.

Вычислены исходная матрица относительных погрешностей элементов матрицы **A** и вектора правой части **y**:

$$\varepsilon = \begin{pmatrix} 0{,}0035 & 0{,}0044 & 0{,}0205 & 0{,}0245 \\ 0{,}0675 & 0{,}0004 & 0{,}0205 & 0{,}0245 \\ 0{,}0552 & 0{,}0732 & 0{,}0003 & 0{,}0245 \\ 0{,}0221 & 0{,}0045 & 0{,}0190 & 0{,}0006 \end{pmatrix}$$

$$\delta = (0{,}0140\ 0{,}0059\ 0{,}0240\ 0{,}0266).$$

На рис. 2 проиллюстрированы результаты расчетов зависимости средних и граничных значений компонент вектора решения от номера компоненты (*а*), а также относительные погрешности отклонения компонент вектора решения по отношению к среднему значению в зависимости от номера компоненты (*б*) в зимней модели.

В работе [11] представлено решение зимней модели межотраслевого баланса двумя способами, включая итерационный метод:

$$\mathbf{x} = ([790{,}77;\ 822{,}71],\ [447{,}07;\ 458{,}07],\ [695{,}01;\ 735{,}18],\ [522{,}83;\ 553{,}01])^{\mathrm{T}},$$

$$\mathbf{x} = ([793{,}7;\ 819{,}6],\ [450{,}4;\ 454{,}6],\ [699{,}8;\ 729{,}2],\ [524{,}7;\ 551{,}2])^{\mathrm{T}}.$$

Вектор, интервальные границы которого x^- и x^+ получены по "знаковой" методике, равен

$$\mathbf{x} = ([795{,}086;\ 827{,}32],\ [449{,}229;\ 460{,}571],\ [698{,}248;\ 738{,}961],\ [524{,}141;\ 554{,}428])^{\mathrm{T}}.$$

Относительная погрешность различия между известными решениями задачи и полученными по предложенной методике не превосходит сотых и десятых процента и не выходит за границы различия двух цитируемых результатов расчетов.

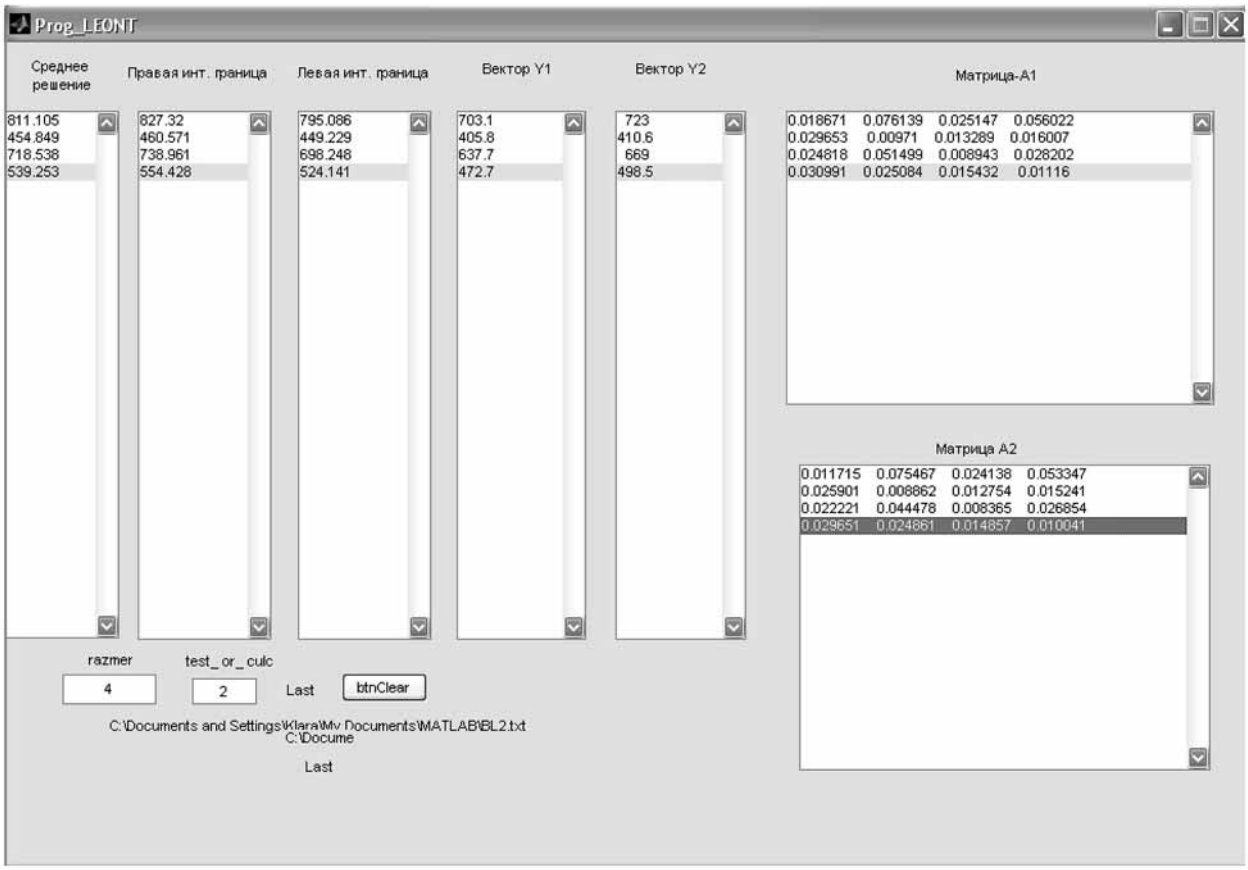


Рис. 1. Листинг интервального вычисления вектора валовой продукции x для зимней модели

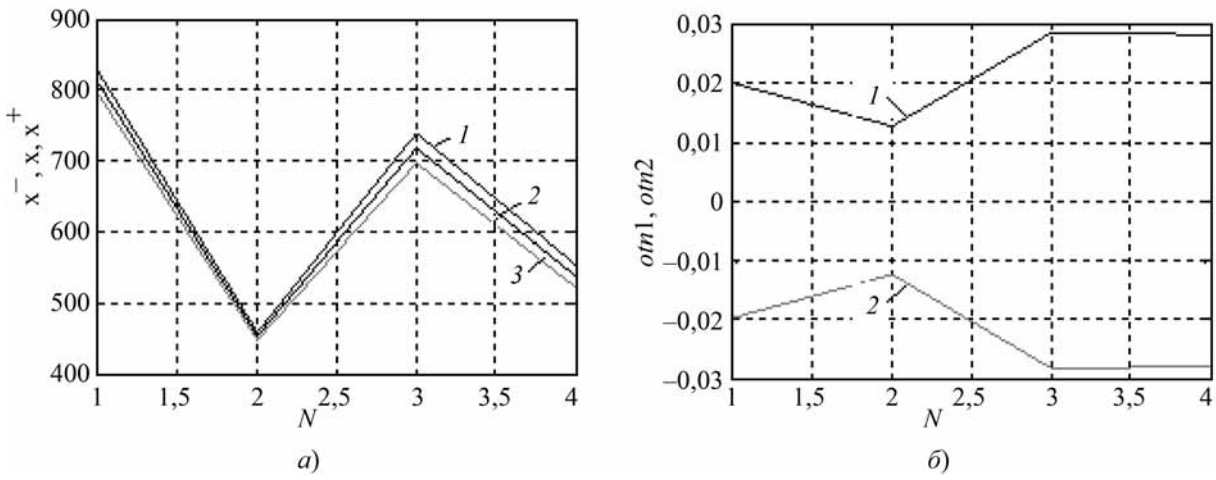


Рис. 2. Зимняя модель:

а — средние значения x компонент вектора **x** (кривая *1*) и интервальных граничных значений x^+ (кривая *2*) и x^- (кривая 3); *б* — относительные погрешности (кривая *1*) и (кривая *2*) границ x_j^- и x_j^+ интервала каждой компоненты $\boldsymbol{x}_j$ к среднему значению x_j соответственно ($otn1 = \frac{\left(x_j^+ - x_j\right)}{x_j}$, $otn2 = \frac{\left(x_j - x_j^-\right)}{x_j}$; N — номер компоненты $\boldsymbol{x}_j$)

Летняя модель:

$$\mathbf{A} = \begin{pmatrix} [0{,}016162;\ 0{,}019384] & [0{,}075185;\ 0{,}075869] & [0{,}025038;\ 0{,}025501] & [0{,}006078;\ 0{,}006468] \\ [0{,}026482;\ 0{,}027443] & [0{,}006179;\ 0{,}007079] & [0{,}013230;\ 0{,}013475] & [0{,}004812;\ 0{,}005121] \\ [0{,}021634;\ 0{,}022443] & [0{,}043191;\ 0{,}050715] & [0{,}008157;\ 0{,}008820] & [0{,}005318;\ 0{,}005660] \\ [0{,}029342;\ 0{,}030214] & [0{,}016439;\ 0{,}016590] & [0{,}015364;\ 0{,}015649] & [0{,}011524;\ 0{,}013679] \end{pmatrix},$$

$$\mathbf{y} = ([739{,}9;\ 750{,}3],\ [415{,}1;\ 420{,}5],\ [642{,}4;\ 656{,}5],\ [324{,}3;\ 346{,}9])^{\mathrm{T}}.$$

На рис. 3 представлен листинг интервального вычисления вектора **x** валовой продукции для летней модели.

В правой стороне на листингах (см. рис. 1 и рис. 3) представлены левые и правые границы A^- и A^+ интервальной матрицы **A** зимней и летней моделей. Первые три столбца справа вверху — точечное решение системы x, правые x^+ и левые x^- границы компонент вектора $\mathbf{x}(x^-, x^+)$, полученные из решения модели, четвертый и пятый столбцы — интервальные границы компонент вектора $\mathbf{y}(\underline{y}, \overline{y})$, задаваемые по условию задачи.

Исходная матрица относительных погрешностей элементов матрицы **A** и вектора правой части **y**:

$$\varepsilon = \begin{pmatrix} 0{,}0016 & 0{,}0045 & 0{,}0092 & 0{,}0311 \\ 0{,}0178 & 0{,}0005 & 0{,}0092 & 0{,}0311 \\ 0{,}0184 & 0{,}0801 & 0{,}0003 & 0{,}0312 \\ 0{,}0146 & 0{,}0046 & 0{,}0092 & 0{,}0011 \end{pmatrix},$$

$$\delta = (0{,}0070\ 0{,}0065\ 0{,}0251\ 0{,}0337).$$

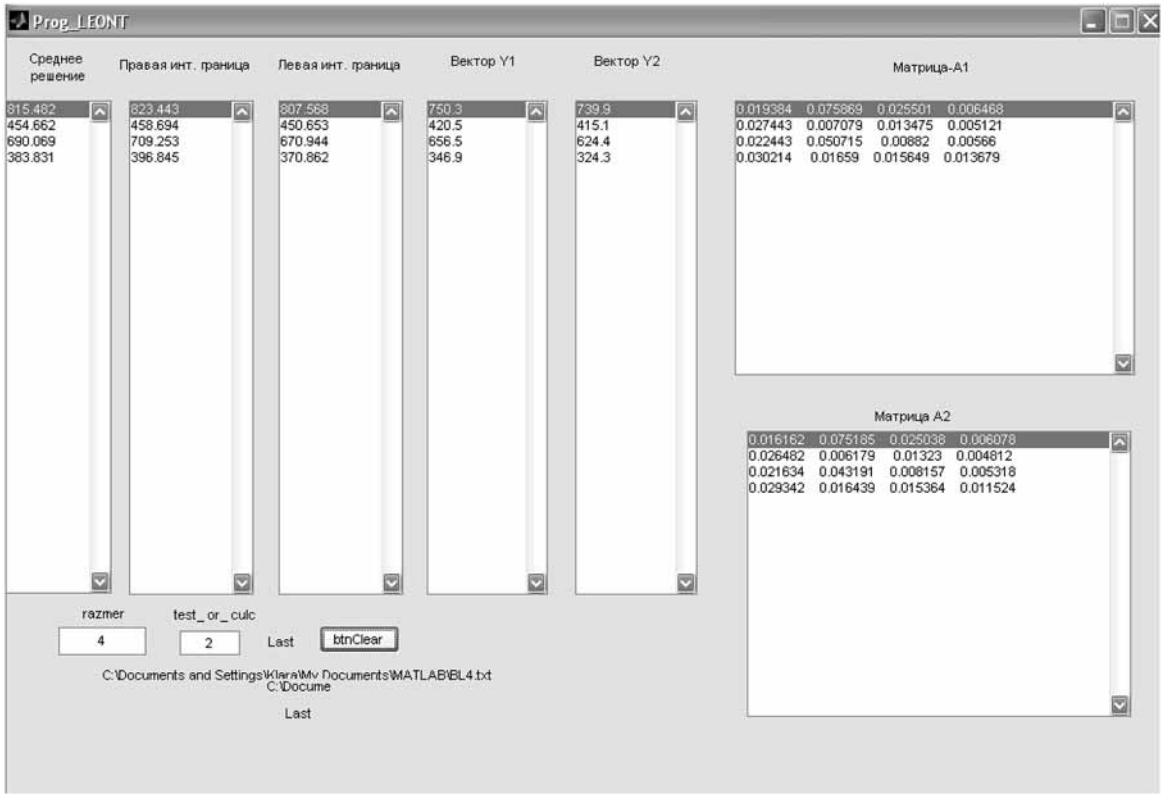


Рис. 3. Интервальное вычисление вектора валовой продукции x для летней модели

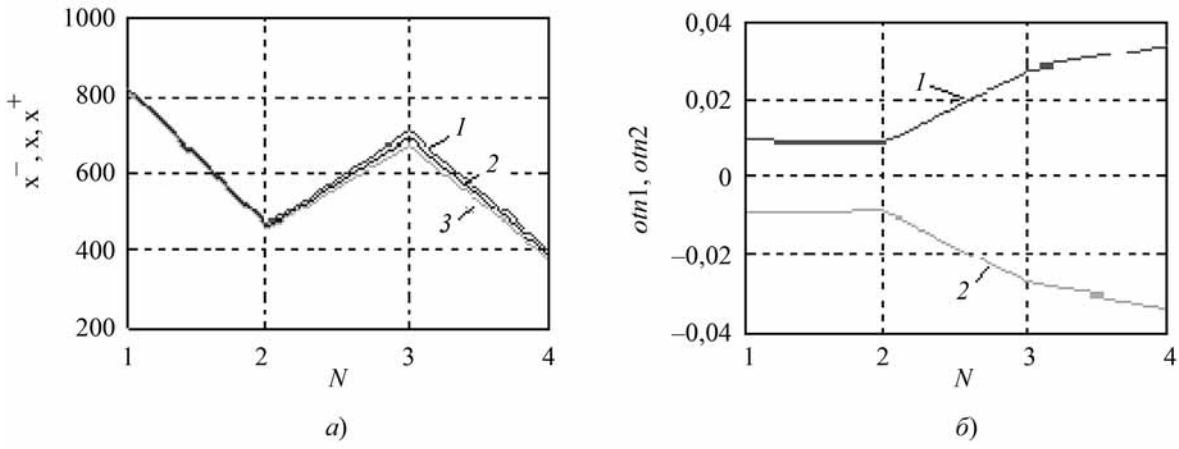


Рис. 4. Летняя модель:

а — средние значения x компонент вектора **x** (кривая 1) и интервальных граничных значений x^+ (кривая *2*) и x^- (кривая *3*); *б* — относительные погрешности (кривая *1*) и (кривая 2) границ x_j^- и x_j^+ интервала каждой компоненты $\boldsymbol{x}_j$ к среднему значению x_j соответственно ($otn1 = \frac{\left(x_j^+ - x_j\right)}{x_j}$, $otn2 = \frac{\left(x_j - x_j^-\right)}{x_j}$; N — номер компоненты $\boldsymbol{x}_j$)

На рис. 4 проиллюстрированы результаты расчетов зависимости средних и граничных значений компонент вектора решения от номера компоненты (*а*), а также относительные погрешности отклонения компонент вектора решения по отношению к среднему значению в зависимости от номера компоненты (*б*) в летней модели.

Решение для вектора **x,** полученное из работы [11]:

$$\mathbf{x} = ([806{,}21;\ 821{,}54],\ [450{,}09;\ 457{,}84],\ [686{,}85;\ 706{,}63],\ [370{,}17;\ 395{,}79])^{\mathrm{T}},$$

$$\mathbf{x} = ([807{,}2;\ 820{,}4],\ [452{,}0;\ 456{,}3],\ [690{,}1;\ 703{,}0],\ [371{,}0;\ 394{,}0])^{\mathrm{T}}.$$

Вектор **x**, интервальные границы которого x^- и x^+ получены по "знаковой" методике:

$$\mathbf{x} = ([807{,}588;\ 823{,}443],\ [450{,}653;\ 458{,}694],\ [690{,}944;\ 709{,}253],\ [370{,}862;\ 396{,}845])^{\mathrm{T}}.$$

На рис. 2 и 4 видно, что относительные погрешности решения, выраженные третьими и четвертыми компонентами x^- и x^+ вектора **x** возрастают до 3 %, но не превосходят среднего значения относительной погрешности для начальных данных, представленных относительными погрешностями коэффициентов и компонент правых частей (15).

Заключение

Продуктивность точечных граничных значений интервальной технологической матрицы $\mathbf{A} = [A^-, A^+]$, проверенных в соответствии с критериями 1 и 2, позволяет воспользоваться упрощенным вариантом оценки допустимого множества решений по "знаковой" методике с использованием положительности обратных матриц $(\underline{E - A})^{-1}$ и $(\overline{E - A})^{-1}$.

Совпадение результатов ранее проведенных расчетов [11] и предложенного метода подтверждает целесообразность подхода по "знаковой" методике к оценке интервального решения и делает обоснованным прогноз в случае, когда входные данные модели имеют неопределенный характер.

Предложенная методика может быть элементарно использована для вычисления систем интервальных линейных алгебраических уравнений большого порядка.

Реализация вычислительных процедур задачи обеспечивается алгоритмом и программой, формализованными *m*-файлами в системе МАТЛАБ [12].

Список литературы

1. **Leontief W. W.** The structure of American Economy 1919—1939. London and New York. Oxford University Press, 1949.

2. **Rohn J.** Input-output model with interval data // Econometrica. 1980. V. 48. P. 767—769.

3. **Шарый С. П.** Алгебраический подход к анализу линейных статических систем с интервальной неопределенностью // Известия РАН. Теория и системы управления. 1997. № 3. С. 51—61.

4. **Шарая И. А.** Строение допустимого множества решений интервальной линейной системы // Вычислительные технологии. 2005. Т. 10, № 5. С. 103—116.

5. **Шарая И. А., Шарый С. П.** Допусковое множество решений для интервальных систем уравнений со связанными коэффициентами // Вычислительные технологии. 2009. Т. 14, № 3. С. 104—123.

6. **Петров Ю. П.** Как обеспечить надежность решения систем уравнений // СПб.: БХВ-СПб. 2009. 172 с.

7. **Иванова К. Ф.** Знаковый подход к оценке решения интервальных линейных систем // Интервальные технологии. 2012. № 9. С. 46—52.

8. **Иванова К. Ф.** Оценка прогнозной модели долгосрочного температурного распределения в деятельном слое почвы // Интервальные технологии. 2013. № 6. С. 54—59.

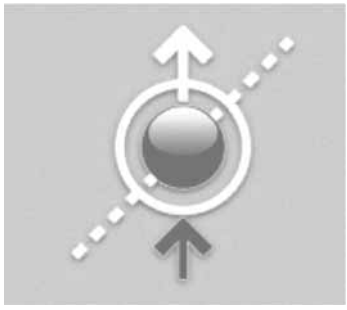
9. **Иванова К. Ф.** Точечно-интервальный метод оценки численного решения уравнений эллиптического типа // Программная инженерия. № 4. 2013. С. 21—28.

10. **Иванова К. Ф.** Чувствительность двойственной интервальной задачи линейного программирования // Программная инженерия. 2013. № 9. С. 24—32.

11. **Горемыкина Г. И., Ляшко М. А.** Избранные разделы линейной алгебры с элементами экономической алгоритмики. Учебное пособие. Балашов, 2005. 95 с.

12. **Иванова К. Ф.** Свидетельство РФ о государственной регистрации программы для ЭВМ № 2013618509 "Программа для оценки бизнес-планирования при распределении ресурсов в экономике и производстве (POKROZ)". 10 сентября 2013 г.

ИНФОРМАЦИЯ



VIII Международная научная конференция "Параллельные вычислительные технологии" (ПаВТ'2014)

31 марта — 4 апреля 2014 г.,
Южный федеральный университет (г. Ростов-на-Дону)

ПаВТ — серия международных научных конференций, представляющих собой авторитетный и престижный форум в области применения параллельных вычислительных технологий в различных областях науки и техники.

Учредителями конференции являются Российская академия наук и Суперкомпьютерный консорциум университетов России.

В первый день работы конференции будет объявлена ***20-я редакция списка Top50*** самых мощных компьютеров СНГ.

Официальный сайт конференции: http://ПаВТ.РФ

ООО "Издательство "Новые технологии". 107076, Москва, Стромынский пер., 4

Дизайнер *Т. Н. Погорелова*. Технический редактор *Е. М. Патрушева*. Корректор *Т. В. Пчелкина*

Сдано в набор 11.11.2013 г. Подписано в печать 19.12.2013 г. Формат 60×88 1/8. Заказ PI114
Цена свободная.

Оригинал-макет ООО "Адвансед солюшнз". Отпечатано в ООО "Адвансед солюшнз".
119071, г. Москва, Ленинский пр-т, д. 19, стр. 1.