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

Саламатин К. М. DiCME — распределенная среда взаимодействия компонентов системы автоматизации экспериментов для физики низких энергий 3

Силаков Д. В. Средства автоматизации поддержки репозиториев программного обеспечения для Linux 12

Французова Г. А., Гунько А. В., Басыня Е. А. Самоорганизующаяся система управления трафиком вычислительной сети: метод противодействия сетевым угрозам 16

Васенин В. А., Роганов В. А., Зензинов А. А. Среда моделирования для исследования средств обеспечения информационной безопасности в Grid- и Cloud-системах 21

Нокель М. А., Лукашевич Н. В. Тематические модели в задаче извлечения однословных терминов 34

Алексеев А. А. Тематический анализ новостного кластера как основа для автоматического аннотирования 41

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

№ 3
March
2014

Published since September 2010

CONTENTS

Salamatin K. M. DiCME — Distributed Components Messaging Environment for Low Energy Physics Experiments Automation 3

Silakov D. V. Automating Maintenance of Linux Software Repositories 12

Frantsuzova G. A., Gunko A. V., Basinya E. A. Self-Organizing Control System of Computer Network Traffic: Method of Counteraction from Network Threats 16

Vasenin V. A., Roganov V. A., Zenzinov A. A. Modeling Environment for Studying of Information Security Solutions in Grid- and Cloud-Systems 21

Nokel M. A., Loukachevitch N. V. Topic Models in the Task of Single-Word Term Extraction 34

Alekseev A. A. Thematic Analysis of a News Cluster as a Basis for Automatic Summarization 41

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

УДК 004.09

К. М. Саламатин, аспирант, Международный университет "Дубна",
e-mail: del@tmpk.ru

DiCME – распределенная среда взаимодействия компонентов системы автоматизации экспериментов для физики низких энергий

Описаны средства автоматического объединения программных компонентов в распределенную систему автоматизации экспериментов (САЭ), их динамического связывания для удаленного выполнения процедур, обеспечивающие возможность использования компонентов в различных экспериментах и системах без изменения остальных составляющих программного обеспечения САЭ. Предложена модель унифицированной программы управления экспериментом, использующей файл описания методики эксперимента, создаваемый пользователем с помощью специальной подсистемы. Файл содержит описания в терминологии пользователя условий регистрации данных вместо обычно используемых названий процедур. Разработанные компоненты ПО САЭ не изменяются при изменении методики эксперимента.

Ключевые слова: *автоматизация экспериментов, распределенные системы, динамическое связывание, сроки разработки, повторное использование*

K. M. Salamatin

DiCME – Distributed Components Messaging Environment for Low Energy Physics Experiments Automation

This paper describes the tools for automatic association of software components into a distributed experiments automation system (EAS), their dynamic binding for remote execution of procedures and ensures the use of components in various experiments and systems without affecting other components of the software EAS. A model of universal experiment control program which uses the file with experiment procedure description is proposed. This file can be created by the user using a special subsystem and contains descriptions of data registration conditions instead of the commonly used procedures names. Developed EAS software components will not be changed when the experimental procedure changes.

The work was performed in accordance with the protocol on the collaboration between the FLNP, JINR and University "Dubna" on the development of the of EAS programming methods.

Keywords: *automation of the experiments, the distributed systems, dynamic binding, development terms, reuse*

Введение

Современные системы автоматизации экспериментов (САЭ) реализуются в виде распределенных приложений, в которых необходима организация взаимодействия между процессами (IPC — *InterProcess Communication*), выполняющимися компонентами программного обеспечения (ПО) САЭ на разных ЭВМ (возможно, и на разнородных процессорах). Взаимодействие сводится к использованию функциональных возможностей одного процесса другим и передаче информации. Важно обеспечить каждому процессу прозрачный способ использования средств IPC. Многие из современных приложений используют для целей IPC технологии RMI, CORBA, DCOM и др. Базовые реализации этих технологий являются, в некотором смысле, расширением RPC (*Remote Procedure Call*), так как позволяют вызвать метод удаленного

объекта и, по сравнению с классическим RPC в конфигурации клиент-сервер, обеспечивают лучшую прозрачность. Однако как и в RPC взаимодействие процессов выполняется по схеме один-с-одним, в некоторых технологиях — синхронно. Для ПО САЭ это является ограничением, так как в системах реального времени события возникают в случайные моменты времени и синхронность взаимодействия приводит к потерям времени на ожидание. Другое и более важное свойство — способ связывания компонентов. Как правило, технологии разработки распределенных систем предоставляют возможность статического и динамического связывания компонентов. Способ динамического связывания компонентов ПО САЭ является критической характеристикой от которой зависит возможность повторного использования компонентов в других экспериментах и разных САЭ. Способ динамического связывания, используемый в популярных технологиях (CORBA, DCOM и др.), для ПО САЭ является избыточно сложными и может быть существенно упрощен, если учесть особенности взаимодействия компонентов в ПО САЭ. Отмеченные выше и некоторые другие соображения стимулировали выполнение разработки специальных средств обеспечения взаимодействия компонентов DiCME (*Distributed Components Messaging Environment*) для использования в ПО САЭ.

Область интересов автора — разработка методов построения ПО для автоматизации экспериментов в области спектрометрии нейтронов на исследовательской ядерной установке ИБР-2 [1] и ускорителе ИРЕН [2]. По этой причине процесс разработки средств обеспечения взаимодействия компонентов DiCME и проектные решения проиллюстрированы на материалах систем автоматизации спектрометрических экспериментов. Однако разработка выполнена в общем виде и пригодна для использования в системах автоматизации экспериментов для физики низких энергий и в некоторых других проблемных областях.

1. Особенности ПО САЭ, влияющие на алгоритмы средств обеспечения взаимодействия компонентов

На основании анализа состава и способа взаимодействия компонентов, используемых в ПО различных САЭ, в работе [3] выделены следующие две основные группы компонентов по их принадлежности к логике приложения:

- базовые компоненты с детерминированным характером взаимодействия, реализующие основную задачу эксперимента (базовую логику) — получение экспериментальных данных, к числу которых относятся интерфейсы пользователя, программа управления экспериментом, программы управления условиями эксперимента и подсистемы регистрации данных (DAQ — ввод, преобразование, архивирование потока данных);
- компоненты (и некоторые методы базовых компонентов), реализующие вспомогательную логику, а именно сервисные функции, обработку нештатных ситуаций и др., не влияющие (в штатных условиях) на выполнение базовой логики ПО САЭ.

Характерные особенности способа взаимодействия группы базовых компонентов следующие [3]:

- схема взаимодействия определена базовой логикой эксперимента и фиксирована, способ взаимодействия один-к-одному;
- в процессе взаимодействия клиент запрашивает выполнение метода и необходима гарантированная доставка сообщений — запроса клиента и ответа сервера;
- в качестве ответа (результата удаленного выполнения процедуры) клиенту нужен только сигнал завершения выполнения запроса; детализирующая информация (например, диагностика — в случае ее наличия) может быть адресована вспомогательной логике;
- нет существенных оснований к тому, чтобы в программу клиента вводить информацию о конкретном связанном сервере (например, адрес) и наоборот; целесообразно использовать слабую связанность компонентов.

Группа компонентов, реализующих вспомогательную логику, может взаимодействовать по другим перечисленным далее правилам [3]:

- схема взаимодействия вспомогательных компонентов не фиксируется, способ взаимодействия один-комногим;
- в составе этой группы два типа компонентов, а именно "источники" и "потребители" информации; в процессе взаимодействия компоненты-источники публикуют информацию, а не вызывают методы потребителей;
- источник может публиковать информацию различного типа (адреса экспериментальных данных, информацию о состоянии узлов установки, диагностические сообщения и др.); потребителю требуется информация определенного типа, потребителей информации любого типа может быть несколько либо ни одного — факт невостребованности этой информации не влияет на выполнение базовой логики эксперимента.

2. Требования к средствам обеспечения взаимодействия компонентов

Средства обеспечения взаимодействия компонентов должны обслуживать все процессы и все варианты их активности в прикладной системе реального времени. В связи с этим средства обеспечения взаимодействия процессов должны иметь следующие свойства:

1) автоматический поиск и динамическое связывание компонентов (автоматический поиск существенно улучшает эксплуатационные свойства системы);

2) асинхронное удаленное выполнение процедур, так как при синхронном вызове снижается скорость выполнения логики приложения (и пропускная способность), а также усложняется программирование системы реального времени, где процессы выполняются одновременно и события, требующие обработки, возникают асинхронно;

3) возможность передать информацию нескольким другим процессам;

4) одна и та же среда взаимодействия должна обрабатывать все обмены в рамках ПО САЭ, так как гомогенную систему намного легче программировать и поддерживать;

5) где бы ни исполнялся процесс, он должен иметь возможность взаимодействовать с любым процессом в системе, используя единый механизм, который не зависит от размещения процессов;

6) возможность перемещения процесса с одной машины на другую (это облегчит устранение аварийных ситуаций, приводящих к выходу из строя ЭВМ);

7) интерфейс компонента не должен зависеть от границ ЭВМ — программы должны быть одинаковы при обращении к процессу на той же машине или на другой, в том числе и другого типа;

8) потеря процесса, ЭВМ или разрыв сетевой связи не должны приводить к разрушению остальной части системы;

9) обеспечивать автоматическую адаптацию к используемой конфигурации.

Реализация средств обеспечения взаимодействия компонентов, отвечающих перечисленным требованиям, упростила разработку, облегчила эксплуатацию и развитие ПО САЭ, повысила надежность работы прикладных систем и эффективность работы пользователей.

3. Проектные решения

Современные программные САЭ строятся из взаимодействующих компонентов, распределенных в сети, при этом функциональность одних компонентов должна быть доступна другим. От способов объединения компонентов в систему и их связывания для удаленного использования процедур других компонентов зависит возможность использования компонентов без изменения в разных экспериментах, сроки разработки и надежность работы САЭ.

3.1. Известные решения

Среда DIM. Приведенному составу требований в некоторой степени соответствует среда обеспечения взаимодействия DIM (*Distributed Information Management*) [4]. Однако эта среда, изначально ориентированная на физику высоких энергий, используется и в других крупных системах (например, в работах [5, 6] приведено описание среды DIM, используемой в системах, распределенных на сотнях ЭВМ и включающих тысячи компонентов). Она имеет следующие особенности:

- введены двухуровневые списки серверов, что является избыточным для задач физики низких энергий;
- заполнение списков серверов (конфигурирование) выполняется разработчиками, в связи с чем требуется персонал сопровождения, как следствие, возникает дополнительный источник ошибок оператора;
- использование списков серверов вносит статическую связанность, в результате реализация среды взаимодействия теряет универсальность и ухудшаются условия повторного использования компонентов; для уникальных систем [5, 6] потеря возможности повторного использования компонентов не имеет значения, но противоречит постановке задачи в данной работе;
- не учитывается специфика ПО САЭ для физики низких энергий, соответственно, не учитывается различие в способах взаимодействия компонентов базовой и вспомогательной логик, и это обстоятельство усложняет развитие прикладной системы, а также уменьшает возможность повторного использования компонентов.

Технология CORBA. Одна из важнейших характеристик средств обеспечения взаимодействия — способ связывания компонентов. Для систем автоматизации экспериментов в области спектрометрии многократное изменение методики эксперимента за время жизни ПО САЭ свойственно самому процессу экспериментальных исследований. Изменение методики исследования сопровождается изменением состава используемых компонентов. Поэтому для САЭ нужен механизм, обеспечивающий динамическое связывание компонентов в соответствии с изменениями логики эксперимента. Наиболее популярная технология CORBA [7] реализует динамическое связывание через специальный набор средств, а именно DII (*Dynamic Invocation Interface*) на стороне клиента и DSI (*Dynamic Skeleton Interface*) на стороне сервера. Через эти интерфейсы для настройки каждого взаимодействия осуществляется ряд вызовов, во время которых выясняются имя метода сервера, типы и значения его аргументов, тип результата, выполняются вызов метода и, наконец, получение результата. В итоге при использовании динамического связывания по технологии CORBA [7] появляются перечисленные далее издержки.

- Для поддержки сценария взаимодействия клиентский и серверный компоненты дополняют избыточными программными фрагментами, не связанным с реализацией логики приложения.
- Среду обслуживания взаимодействия и дополнительные программные фрагменты в клиентском и серверном компонентах привязывают к данной технологии.
- Существенно усложняется программирование реализации как среды взаимодействия, так и компонентов. Реализация динамического связывания требует дополнительно несколько сотен операторов.
- Процесс обслуживания взаимодействия занимает около 130 мс, что в ~40 раз медленнее эквивалентного вызова при статическом связывании [7].

Перечисленные издержки приводят к ухудшению условий повторного использования компонентов, и это обстоятельство является решающим доводом против использования в ПО САЭ связывания по технологии CORBA. Отрицательная оценка опыта использования CORBA при разработке ПО САЭ представлена, например, в работе [8].

В связи с отмеченными выше моментами было принято решение о выполнении разработки распределенной среды обеспечения взаимодействия компонентов, учитывающей специфику задач автоматизации экспериментальных исследований. Рассмотрим возможность иной организации динамического связывания компонентов для ПО САЭ.

3.2. Интерфейсы и логика взаимодействия компонентов ПО САЭ

Для двух групп компонентов (базовых и вспомогательных) можно определить (и фиксировать их состав) варианты интерфейсов. Для базовых компонентов достаточны интерфейсы, обеспечивающие:

— удаленное выполнение метода сервера — передачу запроса на выполнение действия;

— передачу сигнала завершения выполнения действия;

— публикацию информации, адресованной вспомогательной логике.

Для удаленного выполнения метода можно использовать компоненты несколькими различными способами. В CORBA используется вызовов метода. Для этого необходимо выполнить ряд обращений к библиотеке интерфейсов и несколько обращений к диску. Все это выливается в большое число машинных циклов. В работе, результаты которой представлены в статье, вместо вызовов методов используется обмен сообщениями с последующей интерпретацией сообщения в теле сервера. Такой подход дает преимущество в виде относительной независимости запрашивающего и отвечающего компонентов. Компоненты, работа которых основана на обмене сообщениями, слабо связаны между собой и могут на основании содержания сообщений предпринимать нужные действия.

В рамках базового состава компонентов присутствуют компоненты разных типов. Информация о типе компонента (о его функциональном назначении) может варьировать способ поиска компонента и использования интерфейса, а информация о типе сообщения — способ его интерпретации. Таким образом, взаимодействие базовых компонентов может обслуживаться одним интерфейсом, обеспечивающим передачу текстового сообщения, содержащего идентификатор адресата, информацию о типе компонента и типе сообщения. В случае запроса на удаленное выполнение метода сообщение дополняется списком параметров, а после завершения выполнения метода клиенту возвращается сигнал синхронизации, разрешающий продолжение развития базовой логики. Информация, детализирующая результат выполнения метода (например, список зарегистрированных файлов и т. п.), адресуется вспомогательной логике.

В составе компонентов выделены источники информации и ее потребители. Источники вырабатывают, например, следующую информацию:

- сигнал отказа (NACK или ERROR) при приеме сообщения и детализирующую информацию;
- сигнал о возникновении ошибки при выполнении метода и детализирующую информацию;
- информацию о зарегистрированных экспериментальных данных;
- периодически передаваемую информацию о состоянии контролируемых объектов;
- информацию о событии, изменении состояния;
- данные мониторинга объектов и др.

Нет необходимости, чтобы потребители информации имели явного клиента, а источники — конкретный сервер. Источники и потребители должны быть связаны по типу информации. Для компонентов-источников информации также достаточно одного интерфейса, обеспечивающего публикацию информации с указанием ее типа, но без ожидания подтверждения ее получения или использования. Однако потребитель информации при этом может инициировать дополнительные действия, например, для организации цепочки обратной связи.

3.3. Динамическое связывание базовых компонентов

Предлагаемый метод динамического связывания компонентов использует идентификаторы компонентов, в отличие от использования сетевых адресов, принятых в среде DIM [4] или технологиях CORBA, Ice и др.

Реализация этого метода связывания основана на использовании специальной модели программы управления экспериментом и описания методики эксперимента. В соответствии с моделью программы управления экспериментом выполняется два абстрактных процесса и происходит управление их чередованием.

1. Формирование условий, в которых должна быть выполнена регистрация экспериментальных данных.

2. Регистрация данных.

Такой программе управления для конкретизации функционального содержания исполняемых процессов требуется описание условий регистрации и от каждого процесса — сигнал завершения работы, который будет синхронизировать последовательность действий. В работе [9] предложен унифицированный вариант подсистемы описания методики исследования, которая создает файл задания на эксперимент в виде списка описаний условий, в которых должна быть выполнена регистрация данных. В описании условия присутствуют идентификаторы компонентов, списки параметров и их значения, определяющие нужные условия. Идентификатор введен для поиска исполняющего компонента и связывания с ним всегда одного и того же компонента — программы управления экспериментом, а список параметров должен быть передан серверу для интерпретации и выполнения действия. В итоге роль программы управления экспериментом вместо вызова процедур сводится к диспетчеризации используемых компонентов, список которых определяется методикой конкретного эксперимента. Благодаря этому может быть исключен диалог между компонентами, используемый, например, в технологии CORBA для настройки динамического связывания компонентов и удаленного вызова процедур.

3.4. Связывание компонентов вспомогательной логики

Наиболее существенным отличием способа реализации вспомогательной логики от базовой является необходимость передать информацию нескольким процессам, состав которых, вообще говоря, источнику информации неизвестен. Можно решить эту задачу несколькими способами, а именно:

1) поллинг — каждый потребитель периодически опрашивает источник информации и получает ее по готовности;

2) источник информации использует широковещательное сообщение, потребитель опознает нужное сообщение и инициирует взаимодействие;

3) потребитель однократно декларирует интерес к информации определенного типа, после чего специальный компонент обслуживает всех "подписавшихся" потребителей при появлении этой информации.

Очевидно, поллинг невыгоден по причине неоправданного увеличения трафика. Второй и третий механизмы, работающие с использованием прерываний, по крайней мере в два раза уменьшат число сообщений по сети. Помимо этого, обеспечивается возможность параллельной работы, так как клиенту не нужно ждать ответа сервера, и несколько клиентов получают обновления одновременно. Широковещательные сообщения с использованием протокола UDP — достаточно простой способ решить задачу, однако использование UDP не гарантирует доставку. Потеря сообщения вспомогательным компонентом не приводит к нарушению выполнения логики эксперимента. Однако в работе, результаты которой представлены в настоящей статье, выбран третий вариант, обеспечивающий минимальный трафик, гарантированную доставку сообщения и прозрачный алгоритм взаимодействия.

При таком способе связывания компонентов вспомогательной логики:

— возникает возможность динамически включать в ПО САЭ вспомогательные компоненты на любом этапе выполнения эксперимента;

— обеспечивается полная свобода в развитии вспомогательной логики;

— возникновение источников информации нового типа (и соответствующих потребителей этой информации) не приводит к изменению средств обеспечения взаимодействия.

4. Алгоритмы средств обслуживания взаимодействия компонентов

Средства обслуживания взаимодействия процессов компонентов DiCME и библиотека функций должны решать следующие задачи:

- кэширование списка активных компонентов и обновление его с заданной периодичностью;
- автоматический поиск и подключение компонентов к системе;
- динамическое связывание компонентов;
- обеспечение асинхронного выполнения методов сервера;
- передачу синхронизирующих сообщений о завершении выполнения вызванного метода;
- публикацию информации различного типа: о зарегистрированных экспериментальных данных, возникших ошибках, отказах и др.;
- регистрацию подписчиков на предоставление публикуемой информации определенного типа;
- передачу подписчикам информации требуемого типа по мере ее появления.

Для решения этих задач в компонент DiCME включены средства передачи сообщений, блок автоматического формирования и динамической коррекции реестра активных компонентов, брокер сообщений и др.

4.1. Динамическое подключение компонентов к САЭ

Обнаружение компонентов для динамического объединения компонентов в ПО САЭ выполняется при помощи блока, реализующего протокол SLP (*Service Location Protocol*). Данный блок содержит функцию "маяк", которая по запросу компонента рассылает multicast-сообщения, содержащие идентификатор (GUID), тип компонента (например, DAQ, CONDITION) и другую информацию (опционально). Данная информация кэшируется соответствующей функцией блока SLP. Компоненту предоставлен интерфейс, при помощи которого он может получить список типов и идентификаторов компонентов, доступных в данный момент.

Блок SLP периодически опрашивает локальную сеть при помощи multicast-сообщений в целях обновления кэша. В зависимости от требуемого режима работы выполняется настройка блока SLP с помощью ряда параметров. Наиболее важные из них:

- интервал обновления кэша компонентов;
- частота включения "маяка";
- время, в течение которого компонент считается активным (online);
- включение режима форсированного поиска: если запрошенный компонент не числится в реестре активных, будет выполнен поиск его по сети.

Как правило, поиск нужного компонента идет либо по идентификатору (как в случае с поиском компонентов, перечисленных в задании на эксперимент), либо по его типу (например, поиск программы управления экспериментом, представляемой в составе ПО САЭ в единственном экземпляре).

4.2. Механизм обслуживания базовой логики

Механизм обслуживания базовой логики удобно продемонстрировать на примере работы программы управления экспериментом, которая реализует базовую логику, используя файл задания. Структура файла задания [9] показана на рис. 1. Задание на эксперимент содержит описания всех нужных состояний установки, при которых будет выполнена регистрация

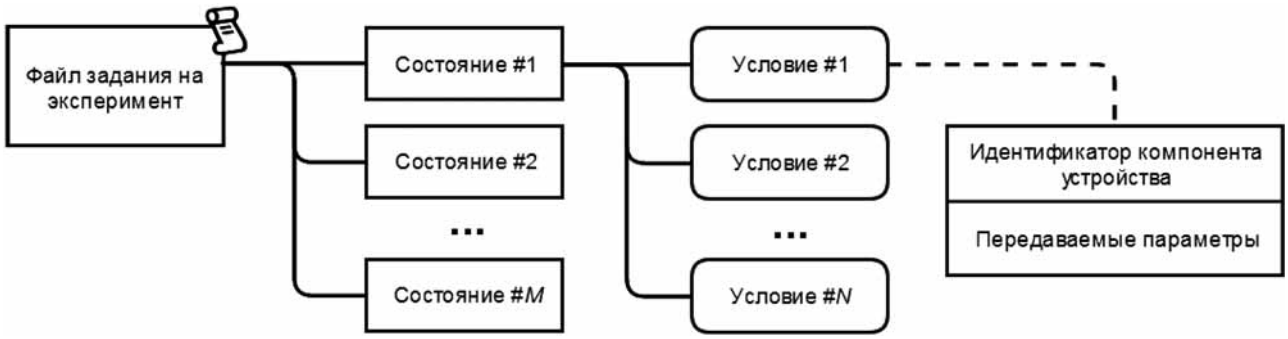


Рис. 1. Структура файла задания на эксперимент

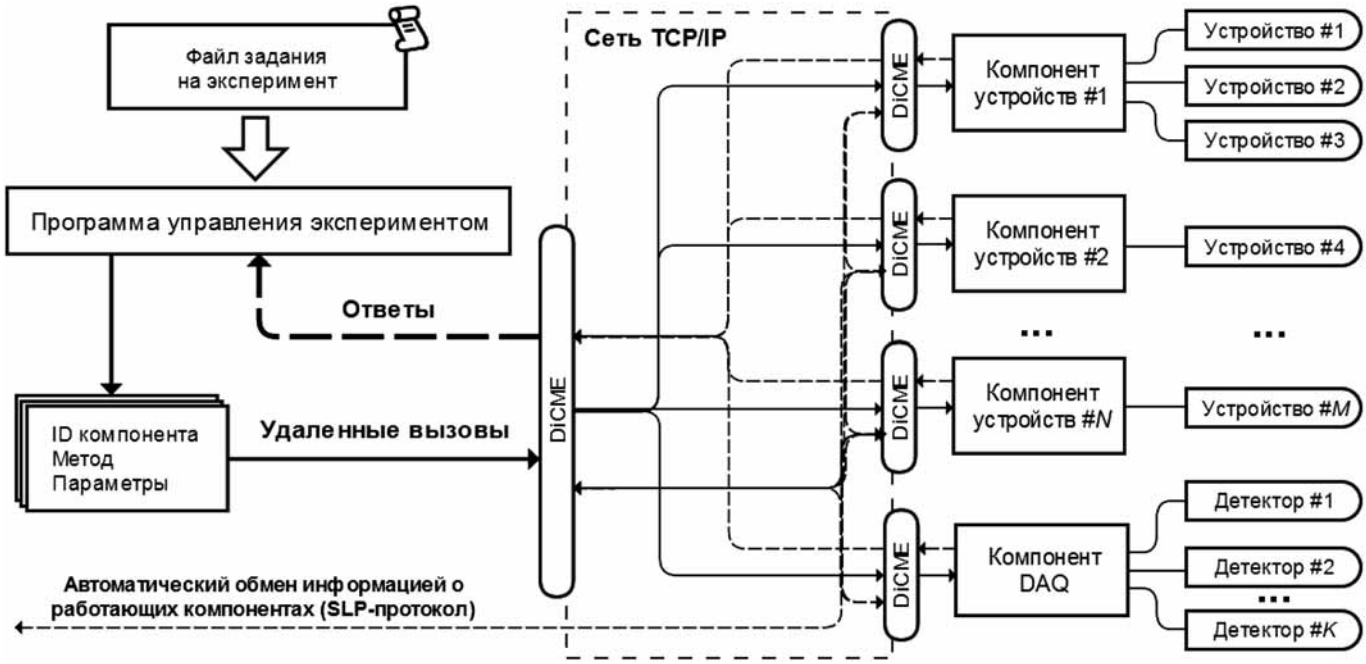


Рис. 2. Схема использование компонента DiCME при реализации базовой логики

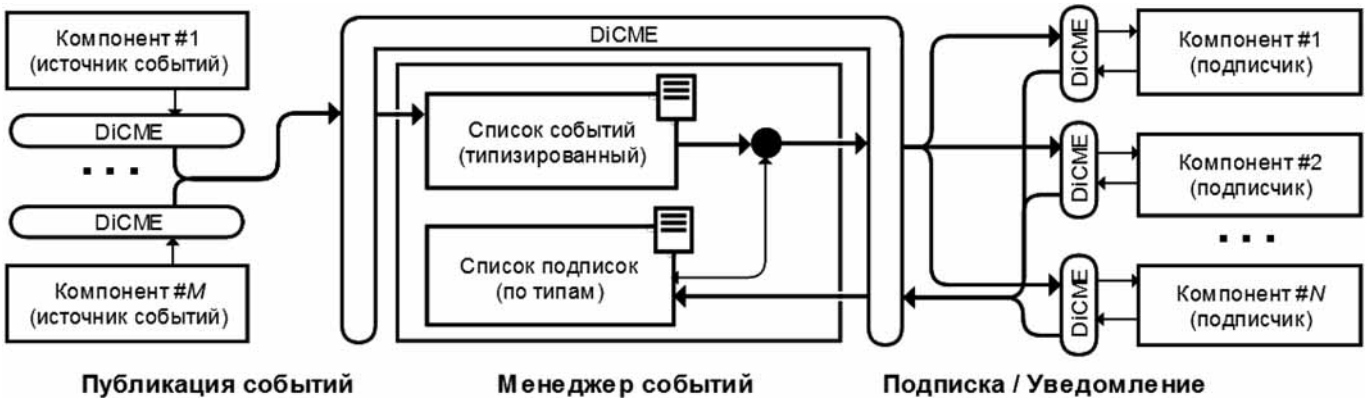


Рис. 3. Схема "мягкого" связывания компонентов вспомогательной логики

```
Const
  // = ID клиента
  ClientGUID:string='aba97325-0312-42a5-8ef5-e9b007f367f3';
  ClientName:string='CliName';                // = Имя клиента
  ClientType:string='CliType';                // = Тип клиента
  ParametersList:string;                      // = Список параметров
//=============================================================
// Подключение и использование DiCME - 4 оператора:
     dicme:=TDiCME.Create(ClientGUID, ClientName, ClientType);
     // Подключение обработчиков событий:
     dicme.OnError  := @ClientOnError;
     dicme.OnResponse    := @ClientOnResponse;
     //-------------------------------------------------------------------
     ----------
     // Использование списка параметров из файла задания:
     ParametersList:= '{device:"polarizer",param:"angle",value:20}';
     //JSON - формат
     // Передача сообщения:
     dicme.Execute(ServiceGUID,'SetParam',ParametersList,False, nil);
//=============================================================
// Обработчики событий - прием ответа:
procedure ClientOnResponse(const Seed: Pointer; Response: String);
begin
  // Текст программы
end;
procedure ClientOnError(const Seed: Pointer; Response: String);
begin
  // Текст программы
end;
//=============================================================
```

Рис. 4. Пример способа подключения к средствам DiCME и использования их возможностей клиентом

экспериментальных данных, и обеспечивает информационную поддержку динамического связывания компонентов (программы управления экспериментом с компонентами управления условиями регистрации данных и подсистемами DAQ).

Схема взаимодействия программы управления и DiCME при выполнении базовой логики показана на рис. 2.

Программа управления экспериментом выбирает описание очередного состояния установки, расчленяет его на описания отдельных условий и передает описания условий компоненту DiCME. DiCME по идентификатору находит нужный компонент и передает ему список параметров. Компонент интерпретирует строку параметров и выполняет нужную процедуру. После установки всех нужных условий программа управления экспериментом включает регистрацию данных. Сигнал завершения регистрации данных дает разрешение программе управления экспериментом перейти к обработке очередного состояния из файла задания. Эти действия выполняются для всех состояний, перечисленных в файле задания.

Программа управления и DiCME прозрачны для списка параметров. Данный метод связывания не ограничивает изменения методики эксперимента, представленной в виде описания конечного автомата. При любых изменениях методики он не затрагивает средства обеспечения взаимодействия DiCME.

В случае, когда ответ исполняющего компонента содержит сообщение NACK или ERROR, среда сообщений дублирует ответ (публикует) по каналу сообщений для вспомогательной логики, представляя детализирующие данные.

4.3. Механизм обслуживания вспомогательной логики

Любой компонент может подписаться на получение информации определенного типа, выполнив обращение к диспетчеру событий (через специальный метод компонента DiCME) с сообщением типа нужной информации. Потребитель может подписаться на произвольное число типов сообщений. Его идентификатор регистрируется диспетчером событий в списках по типам.

Публикуемая информация также регистрируется диспетчером событий. Программа для каждого поступившего типа сообщения разыскивает соответствующий список (если отсутствует — заводит новый) и сохраняет сообщение. Схема работы компонента

```
Const
 ServiceGUID:string='db727048-d3db-4f27-8d5c-0eeb036f6fad';// = ID сервера
 ControllerName:string='Kolhida-Motors';     // = Имя контроллера
 ServiceType:string='DevicesControl';   // = Тип компонента
//============================================================
// Подключение DiCME - 5 операторов:
     dicme := TDiCME.Create(ServiceGUID,ControllerName,ServiceType);
     // Подключение обработчиков событий:
     dicme.OnError  := @ControllerOnError;
     dicme.OnMethod := @ControllerMethod;
     // Регистрация метода передачи сообщений:
     dicme.RegisterRPCMethod('SetParam');
     // Включение "маяка":
     dicme.ChangeBeaconState(True);
//============================================================
// Обработчики событий - прием и обработка сообщений:
procedure ControllerMethod (const MethodName: string; const
ParametersList: string;
                    isNotify: Boolean; var Result: string);
begin
// При возникновении данного события параметр ParametersList в данном
// примере будет иметь значение
{device:"polarizer",param:"angle",value:20}
  // Текст программы
  Result := '{status:"ok"}';                    // JSON - формат
end;
procedure ControllerOnError (const Seed: Pointer; Response: String);
begin
  // Текст программы
end;
//============================================================
```

Рис. 5. Пример способа подключения к средствам DiCME и использования их возможностей сервером

DiCME при реализации вспомогательной логики показана на рис. 3. Использованный во вспомогательной логике механизм связывания является "мягким", так как отсутствует жесткое требование найти потребителя публикуемой информации.

Диспетчер событий выполняет передачу поступившего сообщения потребителям, подписавшимся на сообщения данного типа. При появлении нового сообщения оно передается всем подписавшимся потребителям. При появлении нового потребителя ему передается весь список сообщений интересующего его типа. Списки сообщений типа NACK и ERROR сохраняются в файле протокола при поступлении каждого нового сообщения.

5. Примеры использования средств DiCME

На рис. 4—5 приведены тексты из реализации средствами DELPHI двух примеров, иллюстрирующих способ подключения к средствам DiCME и использования их возможностей клиентом (рис. 4) и сервером (рис. 5).

Любой компонент может подключаться к DiCME и выступать как в роли клиента, так и в роли сервера, используя соответствующие команды.

Заключение

Представленная среда обслуживания взаимодействия процессов является связующим слоем между компонентами. Этот слой облегчает программирование, компоновку и модернизацию программной системы в целом, повышает ее надежность, облегчает условия повторного использования компонентов.

К результатам представленной работы относятся следующие.

- Для систем, у которых базовой логикой приложения управляет один компонент (например, для ПО САЭ), разработан и применен метод динамического связывания компонентов с адресацией компонентов по их идентификаторам. При использовании данного метода связывания не требуется изменение средств обеспечения взаимодействия или взаимодействующих компонентов при изменении базовой и вспомогательной логик приложения.

- Введена классификация компонентов по принадлежности к базовой логике и вспомогательной логике ПО САЭ.
- Использованы различные механизмы динамического связывания базовой и вспомогательной логик: "жесткий" для базовой и "мягкий" для вспомогательной.
- Механизм удаленного выполнения процедур реализуется посредством передачи сообщения с последующей его интерпретацией в компоненте-сервере. Благодаря этому обстоятельству минимизирована связанность компонентов, реализован асинхронный механизм выполнения действий, упрощено программирование компонентов и реализация DiCME.
- Разработанные средства обеспечения взаимодействия компонентов позволяют динамически изменять состав и способ размещения в локальной сети базовых и вспомогательных компонентов, что облегчает восстановление работоспособности системы при отказах, улучшает условия эксплуатации и управления.
- Сняты ограничения на развитие логики эксперимента и вспомогательной логики без дополнительного программирования.
- Разработанные средства обеспечения взаимодействия компонентов инвариантны относительно изменения базовой логики приложения, появления источников сообщений нового типа, появления новых обработчиков событий.

Компонент DiCME и функции средств обслуживания взаимодействия процессов написаны в общем виде, они не связаны с конкретной областью приложения и могут быть использованы в различных системах автоматизации экспериментов и в других проблемных областях.

Работа выполнена в соответствии с протоколом о совместных работах Лаборатории нейтронной физики им. И. М. Франка ОИЯИ и университета Дубна.

Список литературы

1. **Драгунов Ю. Г., Третьяков И. Т., Лопаткин А. В. и др.** Модернизация импульсного исследовательского реактора ИБР-2 // Атомная энергия. 2012. Т. 113, Вып. 1. С. 29—34.

2. **Belikov O. V., Belozerov A. V., Becher Yu. et al.** Physical startup of the first stage of IREN facility // Proc. N 48 of International Seminar "ISINN-17". Dubna. 27—29 May, 2009. Dubna: JINR, 2010. P. 10—16.

3. **Саламатин И. М., Саламатин К. М.** Разработка компонентной САЭ для физики низких энергий на основе использования сетевых технологий. Препринт ОИЯИ Дубна 2013. Р13-2013-74.

4. **Gaspar C., Dönszelmann M.** DIM — A Distributed Information Management System for the DELPHI Experiment at CERN // Proceedings of the 8th Conference on Real-Time Computer applications in Nuclear, Particle and Plasma Physics. Vancouver, Canada. June 1993. URL: http://dim.web.cern.ch/dim/papers/CHEP/DIM.PDF

5. **Gaspar C., Schwarz J. J.** The DELPHI Experiment Control System // Proceedings of the first IEEE International Conference on Engineering of Complex Computer Control Systems. Florida. 6—10 November, 1995. URL: http://www.computer.org/csdl/proceedings/iceccs/1995/7123/00/71230417-abs.html

6. **Franek B.** On-line Experiment Control System for the BaBar Detector at PEP-II at SLAC // Proceedings of the International Conference on Computing in High Energy Physics. Chicago. September, 1998. URL: http://hepwww.rl.ac.uk/franek/BaBar/Talks/CHEP98_Franek.PDF

7. **Динамическая** CORBA. URL: http://kunegin.com/ref3/corba4/7.htm

8. **Flemming S. A., James J. A., Schneider R. et al.** The Open Inspire Architecture for Control, Data Reduction and Analysis // NOBUGS2008 Conference. ANSTO, Australia. 3—5 November, 2008. Paper 134. URL: http://www.nbi.ansto.gov.au/cgi-bin/nobugs2008/overview.ws3

9. **Саламатин К. М**. PSJ — Унифицированная подсистема описания методики эксперимента. Сообщение ОИЯИ Р13-2013-92, Дубна, 2007. 11 с.

ИНФОРМАЦИЯ

УДК 004.457

Д. В. Силаков, канд. физ.-мат. наук, ст. архитектор, ЗАО "РОСА", г. Москва,
e-mail: denis.silakov@rosalab.ru

Средства автоматизации поддержки репозиториев программного обеспечения для Linux

Описана автоматизация трудоемких, но и вместе с тем рутинных задач, стоящих перед разработчиками программного обеспечения для ОС Linux, занимающихся поддержкой обширного набора взаимосвязанных программных компонентов. В качестве иллюстрации приводится поддержка и развитие дистрибутива Linux. Рассматриваются свободные инструменты, используемые в ЗАО "РОСА" для автоматизации обновления программного обеспечения в дистрибутиве и массовых модификаций пакетов при изменениях правил сборки.

Ключевые слова: *Linux, управление пакетами, автоматизация тестирования*

D. V. Silakov

Automating Maintenance of Linux Software Repositories

The paper is devoted to automation of resource-consuming but routine tasks performed by Linux software developers who need to maintain large repositories of interconnected software components. Development of a Linux distribution that uses RPM package format is given as an example. We present the tools used in ROSA Company to automate updates of software components in repositories and perform mass modifications of packages in case of packaging policy changes.

Keywords: *Linux, package management, test automation*

Введение

Природа свободного программного обеспечения (ПО) способствует активному использованию сторонних продуктов, выпускаемых под открытыми лицензиями. Такие продукты применяют в собственных решениях различные компании как в их внутренних проектах, так и в проектах, ориентированных на IT-рынок. Помимо очевидных преимуществ использование стороннего кода имеет и недостатки. В частности, необходимо отслеживать обновления используемых продуктов (как минимум, исправления ошибок и уязвимостей), а также следить за взаимной согласованностью используемых версий различных компонентов.

Хорошим примером крупного проекта, использующего большое число сторонних наработок, является практически любой дистрибутив ОС Linux. Современный дистрибутив содержит тысячи всевозможных программных компонентов — пользовательских приложений, инструментов разработчика, библиотек и вспомогательных утилит. Эти компоненты сильно взаимосвязаны и изменения одного из них может повлиять на сотни других, потребовав их адаптации и пересборки. Одна из основных задач разработчиков дистрибутива — поддерживать репозитории дистрибутива в согласованном состоянии, чтобы пользователь всегда мог установить и начать использовать любое ПО, входящее в систему.

Для удобства разработчиков и пользователей, в большинстве дистрибутивов все программные компоненты помещают в репозитории в виде пакетов, которые представляют собой архивы с собранными файлами приложения и вспомогательной информацией. Такой же подход используется и многими производителями ПО для Linux. Более того, распространение приложений в виде пакетов формата RPM рекомендуется стандартом Linux Standard Base (LSB) [1].

Изменения в репозиториях происходят по двум основным причинам: во-первых, регулярно появляются новые версии тех или иных приложений, а во-вторых, время от времени изменяются правила сборки ПО для дистрибутива (например, правила размещения файлов в системе или работы с конфигурационными файлами).

Процесс внесения изменений в репозитории часто сводится к набору рутинных действий с пакетами. При обновлении программного компонента разработчики обычно пробуют сначала собрать новую версию пакета, не меняя ничего в инструкциях сборки, а просто заменив старый исходный код на новый. Выполнить эти действия для одного пакета — задача несложная, но работа с репозиторием из нескольких тысяч пакетов потребует серьезных затрат времени.

Изменения правил дистрибутива обычно сопряжены с пересборкой большого числа пакетов с минимальными модификациями инструкций сборки. Однако ручное внесение таких модификаций также требует больших временных затрат.

Вместе с тем упомянутые выше задачи являются рутинными по своей природе и могут быть возложены на автоматизирующие их решение инструментальные средства (далее для краткости — инструменты). В данной статье описаны такие инструменты, разработанные и используемые в компании РОСА при создании одноименных дистрибутивов. Рассмотрена организация процесса автоматического обновления пакетов с программными компонентами от сторонних разработчиков, по мере выхода новых версий этих компонентов. Представлены инструменты, используемые для автоматического приведения пакетов в репозитории в соответствии с принятыми политиками сборки. Приводятся статистические данные об использовании в компании РОСА этих инструментов.

Обновление версий программных компонентов

Мониторинг появления новых версий. Первоочередная задача, которую необходимо решить при организации обновлений ПО от сторонних разработчиков — это автоматизация обнаружения самого факта появления новой версии той или иной программы и получение ее исходного кода для последующей сборки.

Практически все программные компоненты, входящие в любой дистрибутив Linux, распространяются под свободными лицензиями и их код доступен для загрузки через Интернет на соответствующих сайтах. Для таких проектов задача обнаружения и загрузки новых версий сводится к мониторингу сайтов разработчиков, выявлению изменений, соответствующих выходу новой версии, и загрузке исходного кода, соответствующего этой версии.

Данные задачи могут быть решены с помощью любого веб-краулера, занимающегося регулярным обходом заданных сайтов. В компании РОСА используется сервис Upstream Tracker [2], специализирующийся на отслеживании выхода новых версий различных программных компонентов для Linux. Заслуживает также внимания инструмент cnucnu, используемый для схожих целей в ОС Fedora.

Сборка новых версий. После того как обнаружен факт появления новой версии некоторого компонента и загружен ее исходный код, этот код необходимо собрать. Сборка пакетов в формате RPM осуществляется инструментом rpmbuild на основе так называемых spec-файлов. Эти файлы содержат: непосредственно инструкции по сборке ПО (задание конфигурационных опций, команды компиляции и линковки); общую информацию о пакете (название, версию, описание, лицензию и т. п.); список файлов, входящих в пакет; скрипты, которые будут выполнены при установке или удалении пакета из системы. В этих файлах также указывается имя файла-архива (одного или нескольких) с исходным кодом, из которого будет осуществляться сборка, а также перечень патчей, которые надо наложить на этот код перед сборкой.

При минорных обновлениях большинства программных компонентов в spec-файле соответствующего пакета достаточно поменять версию и название файла с исходным кодом — инструкции сборки, набор файлов и прочие характеристики пакета остаются неизменными. Именно эти действия и выполняет инструмент Updates Builder при попытке собрать новую версию.

Текущая реализация скрипта сборки новых версий рассчитана на работу со сборочной средой ABF (Automated Build Farm), которая используется для сборки продуктов компании РОСА. Исходный код программных компонентов в ABF хранится в специальном файловом хранилище, а spec-файлы и прочие вспомогательные компоненты (например, специфичные для дистрибутива патчи, накладываемые перед сборкой) — в Git-репозитории.

Перед сборкой новой версии пакета, инструмент Updates Builder создает отдельную ветку в Git-репозитории, клонирует туда текущую рабочую ветку и обновляет spec-файл. После этого осуществляется попытка собрать пакет с новым исходным кодом и новым spec-файлом. В случае успеха собранный пакет помещается в отдельный контейнер без публикации в какой-либо репозиторий.

Мэйнтейнеру пакета (человеку, ответственному за поддержку пакета в дистрибутиве) отсылается письмо о результатах сборки. В случае успеха мэйнтейнер может сразу переходить к проверке функциональности обновленной программы. Если его при этом все устраивает, то он может перенести новую версию из вспомогательной ветки Git в ветку, соответствующую целевому репозиторию. В дополнение к этому, в случае успешной сборки Updates Builder автоматически формирует Pull Request на перенос обновлений в основную ветку Git в веб-интерфейсе ABF. Таким образом, мэйнтейнеры могут быстро просмотреть предлагаемые изменения и согласиться с ними нажатием одной кнопки.

Анализ ошибок сборки. Исправление версии и имени архива с исходным кодом в spec-файле часто является достаточным набором изменений для сборки новой версии пакета. Однако есть несколько часто встречающихся ситуаций, когда необходимо сделать еще ряд тривиальных изменений в spec-файле. К таким ситуациям относятся перечисленные далее.

- **Изменение состава файлов пакета.** В новой версии могут появиться новые файлы, а старые могут исчезнуть или получить новые имена и местоположение. В общем случае необходимо проверить, как эти изменения повлияют на другие пакеты. Однако в большинстве ситуаций подобные изменения не оказывают никакого влияния на другие компоненты дистрибутива, и достаточно просто обновить перечень файлов пакета в spec-файле.
- **Отсутствие необходимости в наложении того или иного патча для новой версии.** Многие патчи, создаваемые для открытого ПО, передаются разработчикам оригинального компонента и внедряются в следующие версии этих компонентов. Соответственно, при обновлении вер-

сии пакета в репозитории некоторые патчи становятся ненужными. Такой подход позволяет минимизировать объем кода, поддерживаемый непосредственно разработчиками дистрибутива, а также избежать дублирования усилий с другими разработчиками, которые, возможно, решают такие же задачи и создают схожие патчи.

- **Нарушение правил формирования пакетов в репозитории.** При сборке новой версии пакета может оказаться, что она нарушает те или иные правила формирования пакетов в репозитории, даже если предыдущая версия этим правилам соответствовала. Например, новые файлы могут оказаться установлены в непредназначенные для них места файловой системы, текстовые файлы с документацией могут содержать переводы строк в стиле Windows и т. д.

- **Требование наличия дополнительных пакетов для сборки.** Новая версия программного компонента может потребовать для своей сборки установки дополнительных пакетов, не требовавшихся для сборки предыдущей версии. Как правило, сборка пакетов в репозитории осуществляется в чистой среде, где установлена только минимальная базовая система и пакеты, явно указанные как необходимые для сборки (естественно, со всеми своими зависимостями). Такой подход позволяет гарантировать воспроизводимость сборки. Однако, если какая-то необходимая зависимость не указана, то сборка завершится ошибкой.

В случае, если сборка новой версии пакета завершилась неудачей, Updates Builder анализирует журнал сборки. Если неудача вызвана одним из факторов, перечисленных выше, то инструментарий пытается автоматически внести необходимые изменения в spec-файл. В этом случае файлы, отсутствующие в новой версии пакета, удаляются из соответствующего перечня в spec-файле, уже наложенные патчи отключаются и удаляются из репозитория, недостающие зависимости сборки добавляются в spec-файл. Для устранения нарушений правил формирования пакетов вызывается инструмент Repo-fixers, описанный в следующем разделе.

После внесения исправлений сборка запускается еще раз. При необходимости, может быть проведено несколько итераций исправления ошибок — ведь по мере решения одних вопросов могут проявляться другие. При этом ведется журнал исправлений для каждого пакета, что позволяет избежать зацикливания. Например, инструментарий может неверно определить имя пакета, который необходимо добавить в перечень сборочных зависимостей. Пакет после такого исправления не соберется, причем ошибка будет та же самая, что и до исправления. В такой ситуации следует прекратить попытки исправить и собрать пакет.

Безусловно, автоматически исправить spec-файл удается не всегда. Например, в случае появления новых файлов может потребоваться мнение человека по поводу того, в какое место файловой системы их устанавливать, а в случае ошибки сборки в силу отсутствия необходимых зависимостей не всегда удается понять, какого именно пакета не хватает. Тем не менее, опыт использования Updates Builder в реальных условиях показывает, что автоматическая доработка spec-файлов на основе анализа журнала сборки позволяет повысить число пакетов, успешно обновляемых в полностью автоматическом режиме, на несколько процентов. При общем числе пакетов в несколько тысяч, такой выигрыш представляется существенным.

Отслеживание соблюдения правил формирования пакетов в репозитории

Поддержка обширного репозитория пакетов существенно упрощается, если все пакеты формируют по одним и тем же правилам. Например, во многих дистрибутивах Linux принято все исполнимые файлы приложений помещать в директорию /usr/bin, библиотеки — в директорию /usr/lib или /usr/lib64 (в зависимости от разрядности системы), документацию — в директорию /usr/share/ doc (в поддиректорию, совпадающую с именем пакета) и т. д. Другим примером является требование явно обозначать пакеты, не имеющие архитектурно-специфичных файлов, указывая архитектуру пакета как "noarch".

Частично подобная унификация пакетов выполняется за счет инструментария сборки [3], однако не всегда можно обойтись без участия человека. В частности, исправление некоторых ошибок возможно только на основе анализа уже собранного пакета. После такого анализа проводится модификация spec-файла пакета и осуществляется повторная сборка. Подобные задачи выходят за рамки функциональных возможностей сборочного инструментария.

Мониторинг репозиториев. Выполнение большинства правил формирования пакетов может быть проверено статически, без установки пакета в систему. Для пакетов формата RPM такой анализ выполняется посредством инструмента Rpmlint, изначально созданного разработчиками дистрибутива Mandrake, а ныне получившего широкое распространение на других системах, в частности, в компании РОСА. Инструментарий Rpmlint обладает гибкой модульной структурой, что позволяет легко адаптировать его под требования конкретного дистрибутива или репозитория. В дополнение к Rpmlint, который работает с отдельными пакетами, в компании РОСА используют утилиты, осуществляющие контроль взаимной согласованности пакетов в репозитории — отсутствие конфликтов по файлам, неоднозначные зависимости и т. д.

В компании РОСА Rpmlint автоматически запускается после сборки каждого пакета, так что мэйнтейнеры могут сразу увидеть возможные нарушения правил дистрибутива. Некоторые нарушения при этом рассматривают как критические и их возникновение ведет к ошибке сборки. Однако многие требования хоть и являются желательными, но их нарушение некритично и допускается публикация в репозиторий пакета, их нарушающего. Тем не менее, по мере возможности (и наличия времени) мэйнтейнеры работают над исправлением таких нарушений. Для облегчения такой работы в компании РОСА настроен регулярный анализ репозиториев с помощью Rpmlint и утилит межпакетного анализа в рамках мониторинга с помощью других средств анализа, описанных в работе [4].

Автоматическое исправление ошибок. Как было отмечено ранее, ряд нарушений правил формирования пакетов может быть исправлен автоматически, однако эти исправления требуют внесения изменений в spec-файл с инструкциями сборки и поэтому не могут быть

осуществлены инструментарием сборки. Для исправления подобных нарушений, в компании РОСА разработан инструментарий Repo-fixers, способный автоматически исправлять некоторые ошибки, выявляемые Rpmlint и сопутствующими утилитами.

Например, при обнаружении в собранном пакете резервных копий каких-либо файлов, либо других артефактов, попавших в пакет по ошибке, инструменты Repo-fixers автоматически добавляют в инструкции сборки пакета команды, удаляющие нежелательные файлы перед созданием архива, помещаемого в пакет. Другими типичными примерами изменений, автоматизируемых с помощью Repo-fixers, являются изменения имен макроопределений, используемых при сборке пакетов, а также избавление от конструкций в скриптах сборки, ставших ненужными — последнее достаточно актуально для компании РОСА, стремящейся сделать сборку как можно более автоматизированной, а инструкции по сборке, поддерживаемые человеком, как можно более короткими и читаемыми.

Мониторинг репозиториев компании РОСА с помощью Rpmlint и исправление ошибок с помощью Repo-fixers проводится на регулярной основе. Такой подход позволяет использовать эти инструменты при изменении правил сборки пакетов для адаптации пакетов к новым правилам и их автоматической пересборки. Для этого достаточно в Rpmlint добавить проверку, обнаруживающую использование старых правил, а в Repo-fixers — код для перевода пакета со старых правил на новые. При следующем запуске Rpmlint найдет все пакеты, которые необходимо перевести на новые правила, а Repo-fixers осуществит такой перевод и соберет исправленные пакеты. Таким образом, разработчикам нет необходимости специально писать скрипты, проходящие по всему репозиторию и вносящие требуемые изменения. Кроме того, использование регулярного мониторинга позволяет обнаруживать и автоматически исправлять регрессии, которые случаются время от времени .

Опыт практического использования

В настоящее время как Updates Builder, так и Rpmlint в связке с Repo-fixers активно используются в компании РОСА при разработке серверной и десктопной линеек одноименных дистрибутивов Linux, а также при сборке родственного дистрибутива OpenMandriva. С помощью Updates Builder осуществляется мониторинг и сборка ~5000 пакетов — преимущественно небольших приложений и утилит, новые версии которых выходят достаточно часто, но которые при этом не требуют какой-либо специальной адаптации для сборки в дистрибутиве. За полгода использования Updates Builder было сделано около 1700 обновлений пакетов, что серьезно сэкономило усилия мэйнтейнеров. При этом около 5 % пакетов были успешно обновлены после автоматического внесения дополнительных изменений в spec-файлы на основе анализа журнала сборки.

Инструментарий Rpmlint на данный момент способен обнаруживать 350 различных нарушений правил формирования пакетов, из которых около 20 исправляются автоматически с помощью Repo-fixers. Малое количество автоматически исправляемых ошибок обусловлено относительной "молодостью" инструментария, а также тем фактом, что большинство ошибок нельзя исправить без участия человека. Тем не менее, для многих ошибок можно автоматически создать заготовку для патча, исправляющего затруднение, и отослать ее мэйнтейнеру, так что последнему необходимо будет провести только те действия, выполнение которых действительно невозможно без его участия.

Заключение

Одной из сильных сторон открытых проектов, к которым относятся и большинство дистрибутивов, является большое число специалистов, принимающих участие в их разработке на безвозмездной основе. Однако в настоящее время усилия многих добровольцев и энтузиастов тратятся на задачи, не привносящие в мир свободного ПО чего-то нового, а на более творческие задачи часто просто не остается ресурсов. По этой причине автоматизация рутинных задач по поддержке репозиториев является актуальным и перспективным направлением. Она позволяет разработчикам сконцентрироваться на действительно интересных задачах и двигаться вперед, а не топтаться на месте.

Отметим, что нередко в отношении подобных автоматизированных инструментов высказывают опасения, что у разработчиков может появиться соблазн полностью переложить процесс обновления пакетов на роботов. Это может негативно сказаться на качестве системы, так как ошибки будут выявляться уже после помещения пакетов в репозитории. В компании РОСА с этим борятся регламентными мерами, путем обновления пакетов в репозитории уже выпущенных релизов, а также разрабатываемых релизов на стадии их бета-тестирования. Такие релизы обязаны проходить через команду Quality Assuarence. В интересах мэйнтейнера перед отправкой обновления на проверку удостовериться в его корректности.

Кроме того, полностью автоматическое обновление с использованием Updates Builder возможно только в случае относительно небольших изменений, которые вряд ли нарушат совместимость с предыдущей версией. В случае серьезных изменений, мэйнтейнерам все равно придется вмешаться, однако инструментарий избавит их от изрядной доли рутинной работы.

Все инструменты, рассмотренные в данной статье, являются открытыми и распространяются под лицензией GPLv2. Инструменты не привязаны к конкретному дистрибутиву и могут быть использованы в любых проектах, требующих поддержки обширных репозиториев ПО для Linux.

Список литературы

1. **Linux** Standard Base Core Specification 4.1. Package Format and Installation. URL: http://refspecs.linux-foundation.org/LSB_4.1.0/LSB-Core-generic/LSB-Core-generic/packagefmt.html

2. **Сервис** Upstream Tracker. URL: http://upstream-tracker.org.

3. **Силаков Д. В.** RPM5: новый формат и инструментарий распространения приложений для ОС Linux // Программная инженерия. 2013. № 7. С. 2—6.

4. **Пономаренко А., Рубанов В.** Задачи и инструменты автоматизации рабочего места майнтейнера операционной системы Linux // Материалы конференции CEEE-SECR 2012. Центр Digital October. Москва. 1—2 ноября 2012. URL: http://2012.secr.ru/lang/ru-ru/talks/problems-and-automation-tools-of-the-workplace-of-a-linux-maintainer

УДК 004.02

Г. А. Французова, д-р техн. наук, проф., e-mail: frants@ac.cs.nstu.ru,
А. В. Гунько, канд. техн. наук, доц., e-mail: gun@ait.cs.nstu.ru,
Е. А. Басыня, аспирант, e-mail: basinya@mail.ru
Новосибирский государственный технический университет

Самоорганизующаяся система управления трафиком вычислительной сети: метод противодействия сетевым угрозам

Обсуждается метод противодействия сетевым угрозам с использованием генетической алгоритмизации и нечеткой логики, который является основой самоорганизующейся системы управления трафиком вычислительной сети.

Ключевые слова: *самоорганизующаяся система управления трафиком, идентификация средств защиты, угрозы сетевой безопасности, фальсификация серверных решений, генетическая алгоритмизация, нечеткая логика*

G. A. Frantsuzova, A. V. Gunko, E. A. Basinya

Self-Organizing Control System of Computer Network Traffic: Method of Counteraction from Network Threats

In this paper we describe the method of the information system protection from network threats by self-organizing control system of computer network traffic, which is based on the genetic algorithms and fuzzy logic.

Keywords: *self-organizing control system of computer network traffic, identification of protection system, network threats, falsification of server solutions, genetic algorithms, fuzzy logic*

Введение

Глобальная сеть Интернет построена на принципах модульности и открытости. С одной стороны это содействует прогрессу, порождая конкуренцию и автономно-параллельное развитие каждого уровня, но с другой стороны это предоставляет базу знаний для проведения злоумышленных действий. Так, зная технологию MAC-адресации сети Ethernet, принципы функционирования протоколов IP, ARP и DHCP, можно провести атаку типа DHCP-liar, организовав поддельный DHCP-сервер, пропуская трафик через необходимый узел и изучая или изменяя информационные потоки. Реорганизация архитектуры локальной вычислительной сети, конфигурирование ее аппаратно-программных узлов и введение криптографических протоколов не могут устранить все уязвимости, и при определенных вычислительных и временных затратах информационные потоки могут быть дешифрованы, а система взломана [3].

Любой бесплатный или коммерческий продукт защиты информации имеет свои уникальные уязвимости, которые в первую очередь определяются запрограммированной "жесткой" логикой.

Исследованиями в области методов обнаружения аномальной активности сетевого трафика и обеспечения информационной сетевой безопасности занимаются российские и зарубежные ученые, например, Р. Н. Селин, R. Lippmann, R. Kwitt, A. Ghosh, В. А. Артамонов, Д. Ю. Гамаюнов, И. М. Ажмухамедов и др. [1, 4, 6, 7]. Но при введении криптографических протоколов либо элементарного дробления пакетов для сокрытия их типа (как, например, организовано в Tor-сетях), данные методы не дают точного результата. В связи с этим возникает необходимость разработки новых методов противодействия сетевым угрозам и обеспечения информационной безопасности трафика предприятия.

1. Угрозы сетевой безопасности

Запрограммированная "жесткая" логика средств защиты и управления позволяет посредством механизмов сканирования и зондирования идентифицировать данное решение и его уязвимости на атакуемом узле. Так же данные инструменты представляют детальную инструкцию по взлому объекта или выводу из состояния доступности.

Следует заметить, что есть категория атак, которую существующие системы защиты и управления трафиком не в состоянии однозначно идентифицировать и отличить от запросов легальных узлов [4]. Примером подобных атак являются распределенные сетевые атаки, один из вариантов которых, "отказ от обслуживания" DDoS, иллюстрирует рис. 1.

Такой тип атаки условно можно разделить на несколько этапов. Сначала злоумышленник заражает N узлов сети Интернет вредоносным кодом, превращая их в "компьютеры-зомби". Затем дает им команду одновременно атаковать одну жертву (какой-либо сервер), например, тактикой полуоткрытых TCP-соединений. Атакованный сервер не может определить, пытается ли установить с ним соединение узел для взаимодействия, либо зараженный хост выводит его из состояния доступности. Необходимо выставление лимитов соединений. Однако для владельца это повлечет непропорциональные затраты времени и потребует вмешательства квалифицированного системного администратора.

В результате информационный ресурс на некоторое время перегружен и недоступен для санкционированных соединений, а возможно и взломан. К сожалению, фильтрация на уровне провайдера, выставление лимитов соединений, введение глобальных "черных" списков адресов зараженных хостов не всегда являются эффективными методами.

Рис. 1. Распределенные сетевые атаки (DDoS)

Следовательно, необходимость защиты информационных процессов и процедуры их электронной автоматизации в области предприятий, корпораций и федеральных государственных учреждений представляет собой актуальную задачу.

Целью работы, описанной в данной статье, является разработка метода противодействия сетевым угрозам, который будет составлять основу самоорганизующейся системы управления трафиком (ССУ), обеспечивающей информационную безопасность локальной вычислительной сети (ЛВС), обладающей свойствами динамической адаптации, оптимизации, автономности с пресечением возможности прогнозирования стратегии реагирования "извне".

2. Метод противодействия сетевым угрозам

Метод противодействия сетевым угрозам, использующий генетическую алгоритмизацию и нечеткую логику, лежит в основе оригинальной системы управления трафиком, работа которой предполагает последовательное выполнение следующих операций.

1. Первоначальное сканирование топологии и конфигурации сети.

2. Конфигурирование базовых правил системы управления трафиком с учетом текущей политики безопасности.

3. Развертывание блоков прогнозирования и фальсифицирования, которое, в свою очередь, можно разбить на ряд следующих этапов:

3.1. установка и конфигурирование изолированных виртуальных серверов;

3.2. анализ установленных экземпляров, формирование выборки;

3.3. подготовка конфигурационных файлов имитации информационных систем для межсетевого экрана (на рис. 2 обозначен буквой *A*, см. третью сторону обложки).

4. Установка для сторонних соединений ССУ модели реагирования, которая выбирается блоком генетической алгоритмизации [5] из выборки, сформированной на шаге 3.2 на определенный интервал времени (итерации по истечению).

5. Динамический сигнатурный анализ трафика с идентификацией попыток сканирования, зондирования и взлома. В случае обнаружения подозрительной сетевой активности системой проводится оценка степени угрозы и предпринимаются, в зависимости от ситуации, следующие действия:

5.1. в случае низкой степени угрозы происходит подключение блока генетической алгоритмизации, состоящей из трех категорий рулеток (объектов, групп и классов объектов), а затем имитация выбранного алгоритмом серверного решения из выборки, составленной на шаге 3.2;

5.2. в случае высокой степени угрозы реагирует блок нечеткой логики ССУ с фальсификацией сер-

верного решения и происходит перенаправление соединения на изолированную серверную модель определенного типа, а также отслеживание дальнейших действий злоумышленника; одновременно с этим проводится и самообучение ССУ с генерацией новых сигнатур (аналогичный метод применяется для имитации состояния "зависания" в целях выявления вредоносных узлов).

6. Трассировка и идентификация хостов-злоумышленников и дальнейшее их внесение в черный список с временной блокировкой.

7. Систематическая самореорганизация системы с предварительной проверкой решений на ее моделях.

Каждый из представленных этапов работы системы управления трафиком включает в себя определенные методы и детализированные алгоритмы действий [8]. Спроектированная таким образом ССУ способна автономно видоизменять существующие и создавать новые алгоритмы, исходя из накопленного опыта и изолированных тестирований на собственных моделях.

3. Инструментарий ССУ

Инструментарием данной системы являются гипервизор XEN, СУБД PostgreSQL, цепочка из трех шлюзов на ОС CentOS (рис. 2, см. третью сторону обложки) с пакетным фильтром iptables и трассировщиком соединений на базе межсетевого экрана Netfilter со сценариями наложения заплат на ядро ОС Patch-o-matic.

Как представлено на рис. 2, ССУ состоит из трех блоков: блока прогнозирования, блока фальсификации серверных решений и корпоративных серверов предприятия на виртуальных машинах. Соответственно, взаимодействуют три межсетевых экрана, главный обозначен буквой *A*.

4. Пример противодействия распределенным сетевым атакам

Рассмотрим в качестве примера реакцию разработанной ССУ на распределенную сетевую атаку (схематично представленную на рис. 1), начиная с этапа развертывания системы. Блок-схема предлагаемого метода противодействия сетевым угрозам показана на рис. 3, где пунктиром выделены основные этапы работы.

В соответствии с описанным методом на этапе 1 происходит сканирование топологии и конфигурации сети с зондированием хостов. На этом шаге идентифицируются задействованные протоколы, технологии и аппаратно-программные средства информационно-технического сектора предприятия. Соответственно, система составляет список уязвимостей (например, доступ к серверу терминалов осуществляется по протоколу RDP 5.* без компонентов-заплат на потенциальную возможность "удаленного выполнения кода") с приоритетом их устранения.

Этап 2 предполагает конфигурирование брандмауэра Netfilter (пакетного фильтра iptables с трассировщиком соединений) с использованием сценариев Patch-o-matic, а именно расширений time, tarpit, random, mport, string, а также других средств. На этом же этапе происходит составление сценария "максимальной защиты" с учетом текущей политики безопасности.

На этапе 3 (развертывание блоков прогнозирования и фальсифицирования) происходит следующее: в пространстве гипервизора виртуальный распределенный сетевой коммутатор разделяет блоки в разные VLAN, делегируя взаимодействие пограничным шлюзам. Системой анализируются установленные экземпляры (серверные ОС), подготавливаются конфигурационные файлы имитации информационных систем.

Этап 4 представлен имитацией модели ССУ для сторонних соединений, выбираемой генетическими рулетками (для объекта, класса и групп объектов) из выборки, сформированной на шаге 3 (см. разд. 2), на определенный интервал времени (итерации по истечению). Проводится синхронизация с глобальными "черными" списками вредоносных сетей (ботнетов), исследование соответствующих вирусных сигнатур блоком нечеткой логики. Выявляются стратегия и типы атак, формируются параметры идентификации угроз и зараженных хостов.

На этапе 5 осуществляется идентификация подозрительной сетевой активности по очереди "полуоткрытых соединений" (началась атака DDoS типа SYN flood), выставление лимита соединений и систематизация информации по ним (блок нечеткой логики исследует типы запросов, параметры дейтаграмм и др.), так как однозначно без априорной информации различить обычные попытки соединения от "троянских" невозможно. Данные систематического анализа трафика и отслеживания подозрительных решений на серверных моделях передаются блоку генетической алгоритмизации (или блоку нечеткой логики в зависимости от степени угрозы), который автономно принимает решение перенаправить информационные потоки данной группы на порты-ловушки (благодаря расширению TARPIT пакетного фильтра IPTABLES) одной из специально подготовленных операционных систем на паравиртуализаторе XEN, отнимая вычислительные мощности атакующих и создавая видимость "зависания" сервера. Это действие провоцирует сервер атакующего прекратить атаку и приступить к попыткам взлома. В ходе этих действий определяется четкий круг атакующих машин.

В соответствии с описанным методом на этапе 6 происходит перенаправление информационных потоков злоумышленника на иную операционную систему на паравиртуализаторе XEN, изолированную от локальной сети и служащую для идентификации хоста злоумышленника и его целей. Проводится балансировка и разгрузка канала с выполнением трассировки (traceroute) до всех зараженных машин с логированием и составлением черного списка.

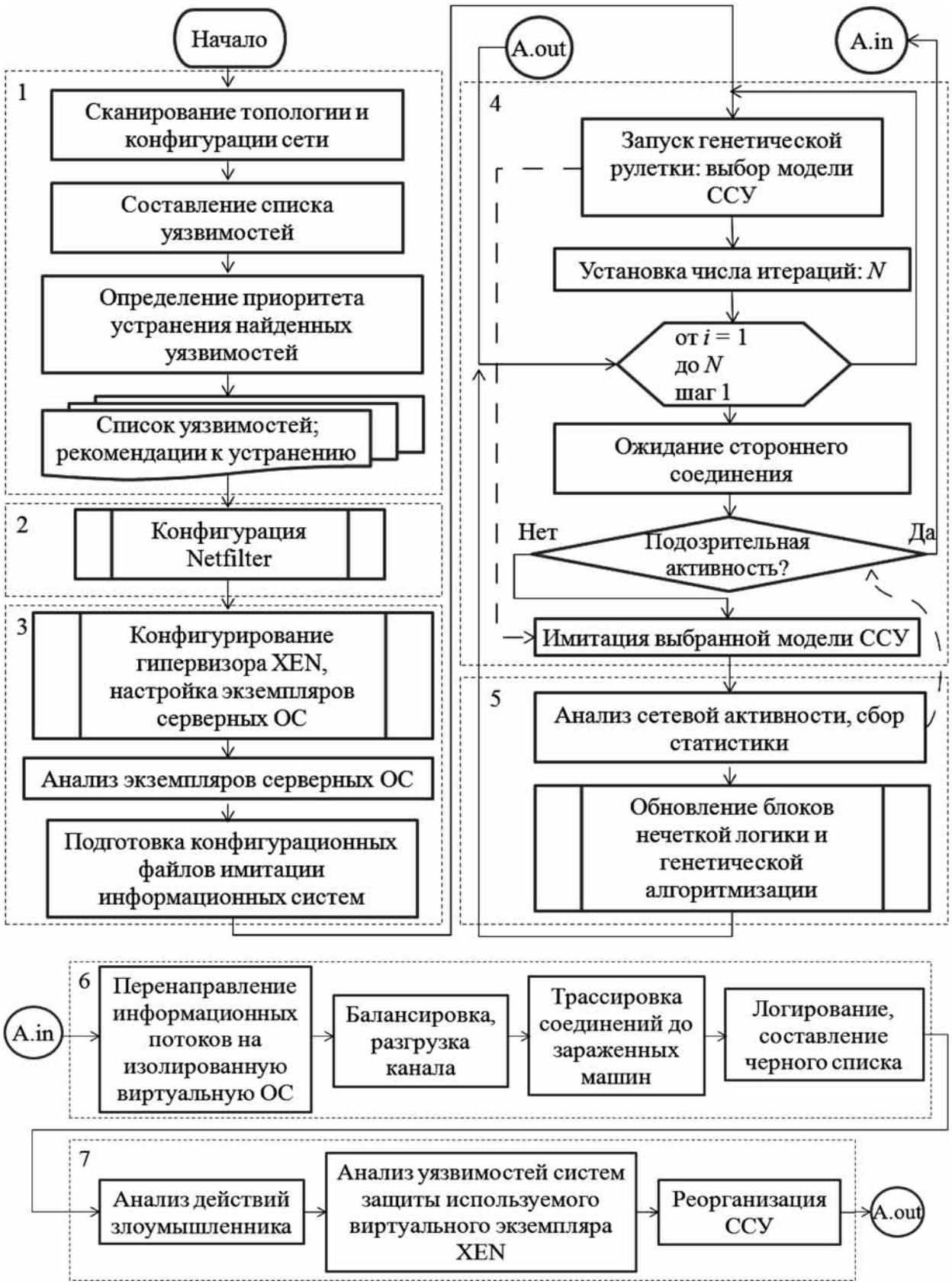


Рис. 3. Блок-схема противодействия распределенным сетевым атакам

Этап 7 предполагает изучение действий злоумышленника и уязвимостей систем защиты, в том числе и операционных систем (например, Windows Server 2012), на которые он был перенаправлен и пытался взломать. По результатам работы происходит последующая самоорганизация ССУ.

Следует отметить, что применение генетических алгоритмов и нечеткой логики в данном случае позволяет работать без большой начальной выборки и выходить на глобальный экстремум решения, минуя локальные, при этом сохраняя достаточную пропускную способность канала [2]. Необходимость проведения быстрого моделирования сложных динамических систем и их сравнительного анализа с заданной степенью точности при нечеткой формализации критериев оценки и сравнения обусловливают использование нечеткой логики.

Таким образом, разработанная система управления трафиком динамически самообучается и самореорганизуется, обеспечивая высокий уровень информационной безопасности с отсутствием возможности прогнозирования стратегии реагирования как с локальной сети предприятия, так и "извне".

5. Результаты экспериментов

Для сравнительного анализа разработанной системы и коммерческих продуктов (Kerio Control 8, Outpost Network Security 3.2, Traffic Inspector 2) проводились эксперименты на идентичной аппаратной основе, а именно Intel® Xeon® E3-1245 Quadcore (8M Cache, 3.40 ГГц) 4 физических ядра, 4 виртуальных (hyper-threading), 16 ГБайт DDR3 ECC, 1 ТБайт SATA 6 Гбит/с 7200 об/м.

Было проведено более 1300 равнозначных распределенных сетевых атак с различными модификациями типов и параметров вторжений. Результаты работы рассматриваемых средств защиты представлены графиком на рис. 4, см. третью сторону обложки.

Согласно результатам проведенного эксперимента из рассмотренного программного обеспечения наилучший результат продемонстрировала предлагаемая ССУ, у которой лишь в 5 % случаев загрузка канала составляла 70 ± 4 %. В рамках проведенных экспериментов попытки взлома и вывода из состояния доступности ССУ были безуспешными.

Заключение

Разработанная самоорганизующаяся система управления трафиком вычислительной сети, которая включает в себя новый метод противодействия сетевым угрозам, хорошо зарекомендовала себя на практике. Она надежно и отказоустойчиво защищает локальную вычислительную сеть, предотвращает перегрузку системы и канала, сохраняя режим недогруженности объектов, оставляя минимум 30 % от предела системных ресурсов и пропускной способности. В результате ССУ, фактически, исключает угрозу информационной безопасности.

Основным ее минусом является требование значительных вычислительных мощностей, в отличие от традиционных средств защиты, для которых достаточно CPU Intel Pentium IV 4 ГГц (или аналогичных), RAM 1 Гбайт, HDD 30 Гбайт. Учитывая, что развитие микроэлектроники стремительно набирает обороты, требование предоставления определенных вычислительных мощностей не является весомым недостатком.

Список литературы

1. **Ажмухамедов И. М.** Динамическая нечеткая когнитивная модель влияния угроз на информационную безопасность системы // Безопасность информационных технологий. 2010. № 2. С. 68—72.

2. **Басыня Е. А., Гунько А. В.** Интеллектуально-адаптивные методы обеспечения информационной сетевой безопасности // Автоматика и программная инженерия. 2013. № 3. С. 95—97.

3. **Басыня Е. А., Французова Г. А., Гунько А. В.** О перспективах развития криптографии // Перспективное развитие науки, техники и технологий. Материалы III Международной научно-практической конференции в 3 томах. Курск: Изд-во ЮЗГУ. 2013. Том 1. С. 199—200.

4. **Гамаюнов Д. Ю., Качалин А. И.** Обнаружение компьютерных атак как задача распознавания образов // Материалы пятого Всероссийского симпозиума по прикладной и промышленной математике. Кисловодск: ТВП, 2004. С. 91—95.

5. **Гунько А. В., Басыня Е. А.** Стохастические методы обеспечения информационной сетевой безопасности // Актуальные проблемы электронного приборостроения. Материалы XI Международной конференции. Новосибирск: Изд-во НГТУ, 2011. Том 7. С. 47—49.

6. **Марьенков А. Н.** Повышение защищенности информационных систем на основе анализа аномалий сетевого трафика // Сборник научных статей 12 Всероссийской научно-практической конференции молодых ученых, студентов и аспирантов. Ярославль: Еще не поздно! 2011. С. 68—69.

7. **Селин Р. Н.** Алгоритм распознавания сетевых атак с мониторингом подозрительной активности и ретроспективным анализом // Известия высших учебных заведений. Северо-Кавказский регион. Технические науки. 2006. Приложение № 1. С. 15—20.

8. **Французова Г. А., Гунько А. В., Басыня Е. А.** Обеспечение информационной безопасности внутренних информационных потоков корпоративной сети // Наука. Технологии. Инновации. Материалы всероссийской научной конференции молодых ученых в 10 ч. Новосибирск: Изд-во НГТУ, 2013. Часть 2. С. 41—43.

УДК 004.7 + 004.056 + 004.94

В. А. Васенин, д-р физ.-мат. наук, проф., зав. отделом,
Институт проблем информационной безопасности МГУ, e-mail: vasenin@msu.ru
В. А. Роганов, ст. науч. сотр., **А. А. Зензинов,** аспирант, программист 2 кат.,
НИИ механики МГУ

Среда моделирования для исследования средств обеспечения информационной безопасности в Grid- и Cloud-системах

Предлагается подход к проведению натурного, имитационного, виртуального и аналитического моделирования, а также их объединения в рамках гибридного режима моделирования для оценки уровня защищенности ресурсов распределенных информационных систем на основе Grid- и Cloud-технологий. Такой подход позволяет исследовать поведение распределенных систем с разных сторон, учитывая особенности архитектуры, программного обеспечения, назначения систем и условий их эксплуатации. В статье представлено описание разработанного программного комплекса для проведения исследований, описаны пути его расширения и способы работы с программным интерфейсом среды моделирования.

***Ключевые слова**: грид-вычисления, облачные вычисления, распределенные вычислительные системы, информационная безопасность, моделирование, натурное моделирование, имитационное моделирование, виртуальное моделирование, гибридное моделирование*

V. A. Vasenin, V. A. Roganov, A. A. Zenzinov

Modeling Environment for Studying of Information Security Solutions in Grid- and Cloud-Systems

In the security evaluation of security level of distributed information systems based on Grid- and Cloud-computing technologies it seems appropriate to model the studied systems and their behavior under different working conditions. The paper proposes an approach to conducting a full-scale, simulation, virtual and analytical modeling, as well as their combinling — hybrid mode simulation. Such approach allows to investigate the behavior of distributed systems from different perspectives, taking into account features of the architecture, software, purposes of these systems and operation conditions. The paper also describes the developed modeling software, the way of its extension and how to work with API of modeling environment.

***Keywords:** grid-computing, cloud-computing, distributed computer systems, information security, modeling, full-scale modeling, virtual modeling, simulation, hybrid modeling*

Введение

В последние годы все большее внимание уделяется программным средствам обеспечения безопасности ресурсов больших распределенных информационно-вычислительных систем. Несмотря на то, что для Grid- и Cloud-систем создано значительное число разноуровневых средств такого назначения, обеспечение их надежной защиты является очень непростой, трудоемкой и ресурсозатратной задачей. Для ее эффективного решения необходимо оценивать и корректировать уровень защищенности подобных систем на разных архитектурных уровнях их реализации. Это обстоятельство побудило авторов спроектировать и реализовать программную среду для гибридного моделирования Grid- и Cloud-систем, при помощи которой в рамках одной модели могут быть одновременно эффективно представлены и опробованы сразу несколько уровней системы при небольших затратах на сам процесс исследования.

Подход к гибридному моделированию уже хорошо зарекомендовал себя и активно используется, в частности, в космической отрасли, позволяя изучать сразу большое число различных режимов поведения сложной системы без реального запуска космического аппарата. В случае Grid- и Cloud-систем для построения гибридной модели были выбраны натурное, имитационное, виртуальное и аналитическое виды моделирования, что и дало по первым буквам перечисленных слов сводное название предлагаемой среде моделирования — НИВА.

Среда моделирования НИВА не предполагает высокой квалификации ее пользователей. В состав среды входят редакторы с графическим интерфейсом для наглядного определения конфигураций проектируемых распределенных систем и сценариев их использования, поддерживается автоматизированное развертывание конфигураций и моделирование заданных активностей как в интерактивном режиме с графической визуализацией, так и в неинтерактивном режиме, управляемом с помощью командных файлов на стандартном интерпретируемом языке. Сценарии работы распределенной системы компонуются на основе библиотеки шаблонов поведения и позволяют компактно описывать как штатную работу распределенной системы, так и ее работу в условиях проведения различных атак.

Таким образом, среда НИВА позволяет с относительно небольшими ресурсозатратами проводить моделирование разрабатываемых Grid/Cloud-систем в целях получения их характеристик в части производительности и информационной безопасности, а также поддерживает их оптимизацию при помощи многовариантного неинтерактивного моделирования.

Существующие подходы

Сама по себе тема моделирования распределенных вычислительных систем не нова и имеет многолетнюю историю. Вскоре после появления распределенных вычислений были созданы средства для их изучения на основе дискретно-событийного моделирования, которое, по мнению авторов, является наиболее универсальным подходом и прекрасно зарекомендовало себя в области изучения сетей передачи данных.

На настоящее время наибольшую известность и широкое распространение получили специализированные средства SimGrid [1], GridSim [2], CloudSim [3], при помощи которых было проведено немало научных, а также чисто практических исследований, призванных объяснить наличие "узких мест" в конкретных распределенных системах. Среди российских средств аналогичного назначения следует отметить систему, разработанную в российском Институте системного программирования РАН [4]. В этих средах распределенная система, как правило, представляется совокупностью входящих в нее объектов верхнего уровня, т. е. вычислительных узлов, межузловых соединений и т. д.

С широким распространением поддержки виртуализации в аппаратуре связан и активно развивается подход на основе виртуального моделирования. В этом случае в модели имитируется аппаратурный уровень при помощи системы виртуальных машин, связанных между собой виртуальными же сетями.

Конечно, есть и другие подходы, например, аналитическое и натурное моделирование. Если при моделировании ставится цель охватить большое число узлов (например, моделировать большие сегменты Интернет), и при этом интересует макро-уровень, то можно попытаться использовать аналитические методы из теории систем массового обслуживания, сети Петри и т. д. В отличие от них натурное моделирование поможет адекватно изучать даже такие тонкие явления в программном обеспечении, как уязвимости типа Race Condition. Каждый из перечисленных подходов хорош на своем уровне, однако ни один из них не способен во всей полноте, эффективно охватить весь спектр возможных взаимодействий, которые имеют место в больших распределенных системах.

Принимая во внимание изложенные выше соображения, представляется целесообразным попытаться объединить все эти подходы. Кроме того, упомянутое ПО для моделирования Grid- и Cloud-систем обусловливает наличие определенной квалификации программиста для установления и анализа интересующих активностей. Однако большинство существующих систем при этом не имеет даже графических средств для представления сетевой конфигурации.

По этой причине главными приоритетами при разработке среды НИВА были поддержка гибридного режима моделирования и удобство работы оператора.

Функциональное назначение

Среда моделирования Grid/Cloud-систем, именуемая НИВА, позволяет проводить натурное, имитационное, виртуальное и аналитическое виды моделирования распределенных вычислительных систем как в интерактивном, так и в неинтерактивном режимах, с возможностью визуализации и специализированной пост-обработки результатов. Сочетание различных моделей позволяет проводить комплексное исследование больших распределенных систем, а также погружать их в реалистичный контекст глобальной сети, а именно:

- при помощи натурного моделирования можно оценить реальную вычислительную эффективность счетных узлов и используемого интерконнекта на реальном оборудовании;
- при помощи виртуального моделирования можно провести необходимые эксперименты с дистрибутивом, оценить совместимость ПО, провести эффективное тестирование на известные уязвимости;
- при помощи имитационной модели возможно смоделировать поведение значительного числа узлов Grid/Cloud, а также последствия возможных атак и динамику распространения скрыто внедренных программ типа вирусов;

• аналитическое моделирование позволяет легко моделировать Grid-систему, как составную часть Интернет, задав при помощи формул вероятности внешних воздействий и интенсивности приходящих из глобальной сети запросов.

Цикл работы оператора состоит из выбора/редактирования конфигурации, выбора/редактирования сценария в соответствующих графических редакторах, загрузки конфигурации перед запуском сценария (их связывание происходит динамически по именам узлов) и запуска процесса моделирования с визуализацией, либо без таковой с возможным последующим детальным анализом полученных результатов. Процесс интерактивного моделирования может быть в любой момент приостановлен оператором (например, для детального изучения состояния моделируемой системы), выполняться по шагам (в терминах квантов виртуального времени), либо досрочно прекращен (например, для редактирования конфигурации). Существует также возможность запрограммировать автоматическую [при]остановку при возникновении определенных условий.

Результатом серии проведенных запусков в неинтерактивном режиме является отчет, на основании которого может быть принято решение о пригодности или непригодности предлагаемых средств и стратегий защиты, а также выработаны рекомендации для дальнейшего повышения надежности и защищенности разрабатываемых или реальных Grid/Cloud-систем.

Разумеется, любой среде моделирования присущ и ряд ограничений. Ограничения на размер моделируемых систем диктуются доступным объемом оперативной памяти, а также быстродействием используемой аппаратной платформы. Однако преимущества гибридного подхода позволяют иногда обходить данный недостаток, предоставляя возможность для аналитического моделирования отдельных подсистем на основе имеющихся о них эмпирических или статистических данных. Как следствие, гибридный подход позволяет охватить практически все случаи рассматриваемых Grid/Cloud-систем ценой потери точности.

Логическая структура

Структурно-программная среда НИВА представляет собой связку ключевых компонентов для решения следующих задач:

1) определение конфигурации и сценария использования Grid/Cloud-систем;

2) преобразование (трансляция) конфигурации и сценария в низкоуровневую программу на языке С для запуска под управлением моделирующего ядра;

3) запуск результирующей программы в составе одного или нескольких симуляторов;

4) визуализация и протоколирование существенных происходящих событий;

5) пост-обработка результатов и выполнение последующих команд оператора (интерактивный режим) или управляющего командного файла (неинтерактивный режим) с переходом к п. 1.

Гибридное моделирование

Общим знаменателем между различными симуляторами в данном случае выступает имитационная модель как наиболее универсальная. Такой подход избавляет от необходимости отвечать на сложный вопрос о прямом взаимодействии множества разнородных симуляторов. Он требует лишь минимальных усилий по созданию "имитационных представителей" для тех сущностей, которые должны взаимодействовать напрямую, но располагаются в разных симуляторах. В качестве базового симулятора для имитационного моделирования используется современный, активно развивающийся фреймворк SimGrid с открытым исходным кодом, дополненный разработанными средствами для интерактивной визуализации, графическим редактором сценариев и библиотекой шаблонов, реализующих типовые активности в Grid/Cloud-системах.

Гибридное моделирование позволяет задействовать одновременно несколько симуляторов. Организовать взаимодействие симуляторов можно по-разному, в рассматриваемой системе было принято решение сделать это на базе имитационной модели.

Механизм этого взаимодействия выглядит следующим образом.

1. Один из имитационных симуляторов (в нашем случае SimGrid) является главным, и все сущности (за исключением моделируемых аналитически) имеют представление как узлы в имитационной модели. Но некоторые также имеют представление в виде реальных процессов на виртуальных машинах или реальных узлах в случае виртуального или натурного моделирования соответственно.

2. Если узел помечен как выполняющийся на внешнем симуляторе, то вместо типовой логики вызова действий SimGrid занимается лишь переназначением задач во внешний симулятор и наоборот. При этом внешний симулятор общается с SimGrid по несложному коммуникационному протоколу.

3. Внешние симуляторы получают полную конфигурацию системы с назначенными действиями. Поэтому они либо обращаются к другим узлам "натурным" образом (т. е. проводят реальные DDoS-атаки, пересылают реальный вредоносный код), либо пересылают сообщение в SimGrid, если целевой узел оказался лишь имитацией.

4. Поскольку в случае натурного и виртуального моделирования действия не имитируются, а выполняются по-настоящему, поддержка этих действий требует реализации сопрягающей логики. То есть при получении сообщения с вредоносным кодом от имитационной модели в задачу натурного симулятора входит ее преобразование в реальный вредоносный код. Поэтому во многих случаях может оказаться проще проводить независимое моделирование в этих моделях и затем сравнивать полученные результаты для повышения уровня достоверности.

Ключевые программные компоненты

Графический редактор конфигураций Eclipse/SimGrid. Редактор конфигураций Grid/Cloud-систем входит в состав используемого базового пакета SimGrid и позволяет задавать и модифицировать типовые конфигурации Grid/Cloud-систем при работе в интегрированной среде разработки Eclipse. Среда Eclipse традиционно используется в качестве оболочки во многих средствах моделирования, а также включает в себя средства для коллективной работы разработчиков (имеет поддержку для работы с сетевыми репозиториями). На рис. 1 показаны примеры задания конфигураций систем, соблюдающих топологию "звезда" (*а*) и "дерево" (*б*).

Графический редактор сценариев. Редактор сценариев (средство определения активностей в Grid/Cloud-среде) имеет простой графический интерфейс и позволяет определить сценарий использования распределенной системы как совокупность параллельно работающих распределенных активностей. Под активностью здесь и далее понимается совокупность взаимодействующих процессов в распределенной системе, сообща решающих какую-либо задачу. Примерами активностей могут служить взаимодействующие пользовательские процессы в ходе выполнения вычислительного задания, а также совокупность вредоносных действий злоумышленника при проведении спланированной атаки. На рис. 2 в левой части представлен список существующих активностей, в правой части — список шаблонов действий (составных частей активностей), редактирование же выбранной активности осуществляется в центральной части.

Активности состоят из действий. Некоторые действия являются подготовительными, т. е. выполняются перед началом моделирования. Это очень важный момент, существенно упрощающий работу оператора, так как вместо частого изменения конфигурации достаточно видоизменить (или деактивировать) активность, незначительно модифицирующую конфигурацию распределенной системы перед стартом.

Графический интерфейс редактора сценариев позволяет также осуществлять просмотр развернутой конфигурации (рис. 3), не дожидаясь запуска на моделирование.

Графический редактор сценариев реализован в виде программы на языке Java при помощи набора стандартных технологий Swing, JavaBeans и с использованием хорошо зарекомендовавших себя библиотек. Для упрощения работы оператора многие операции по редактированию сценария выполняются перетаскиванием объектов при помощи мыши.

Сервер визуализации "Сцена". Для интерактивной визуализации процесса моделирования разработан сервер "Сцена", выступающий как универсальное средство отображения состояния моделируемой системы. Используемый векторный формат SVG поддерживает масштабирование и позволяет отображать сложные совокупности графических элементов с возможностью их выборочной детализации (рис. 4).

Сервер "Сцена" поддерживает концепцию "визуальных переменных". К их числу относят параметры, которые изменяются во время моделирования задействованными симуляторами и отображаются сервером "Сцена" непосредственно при их изменениях (загруженность/доступность узлов, поражение узла вирусом и т. д.). Еще одной особенностью сервера "Сцена" является возможность его запуска на отдельном мониторе (проекторе) в качестве серверного процесса с возможностью параллельного обслуживания нескольких симуляторов.

Сервер визуализации "Сцена" реализован на языке Java и позволяет группе процессов отображать состояние моделируемой системы. При этом один из этих процессов заявляет себя как главный и устанавливает начальную картинку, передавая URI файла с графикой в формате SVG, в котором может присутствовать логика обработки запросов на отрисовку изменений на языке JavaScript. Каждый процесс, участвующий в ходе моделирования, будь то процесс-симулятор или реальный процесс в Grid/Cloud-среде, может посылать запросы на отрисовку отдельных графических признаков (изменение цвета, надписей и т. д.), а также получать информацию об изменениях на сервере "Сцена", что может быть также востребовано в ходе более продвинутой интерактивной визуализации.

Система виртуального моделирования. Виртуальное моделирование основано на работе с виртуальными объектами, такими как виртуальные машины и виртуальные сети. Для управления этими объектами используется хорошо зарекомендовавшая себя кроссплатформенная библиотека libvirt.

Система развертывания представляет собой комплекс программ, написанных на языке Python. Она реализует следующие функции:

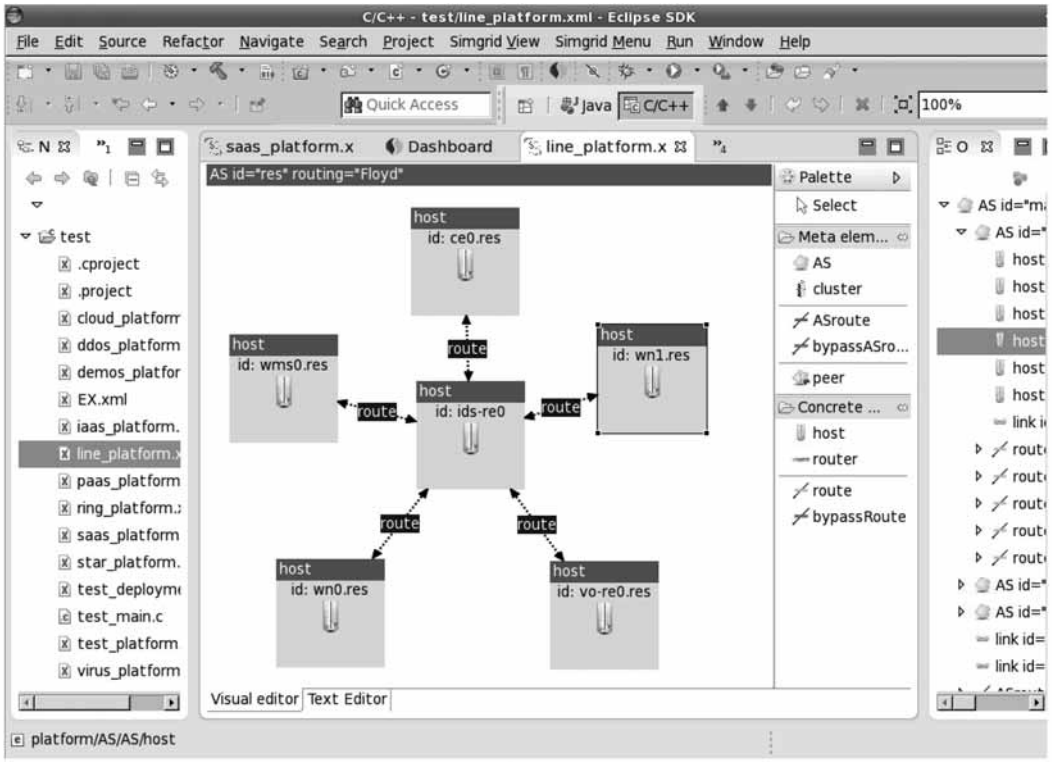
— развертывание виртуальных машин по заданной конфигурации;

— запуск виртуальных машин с сетевым сервисом для выполнения заданий;

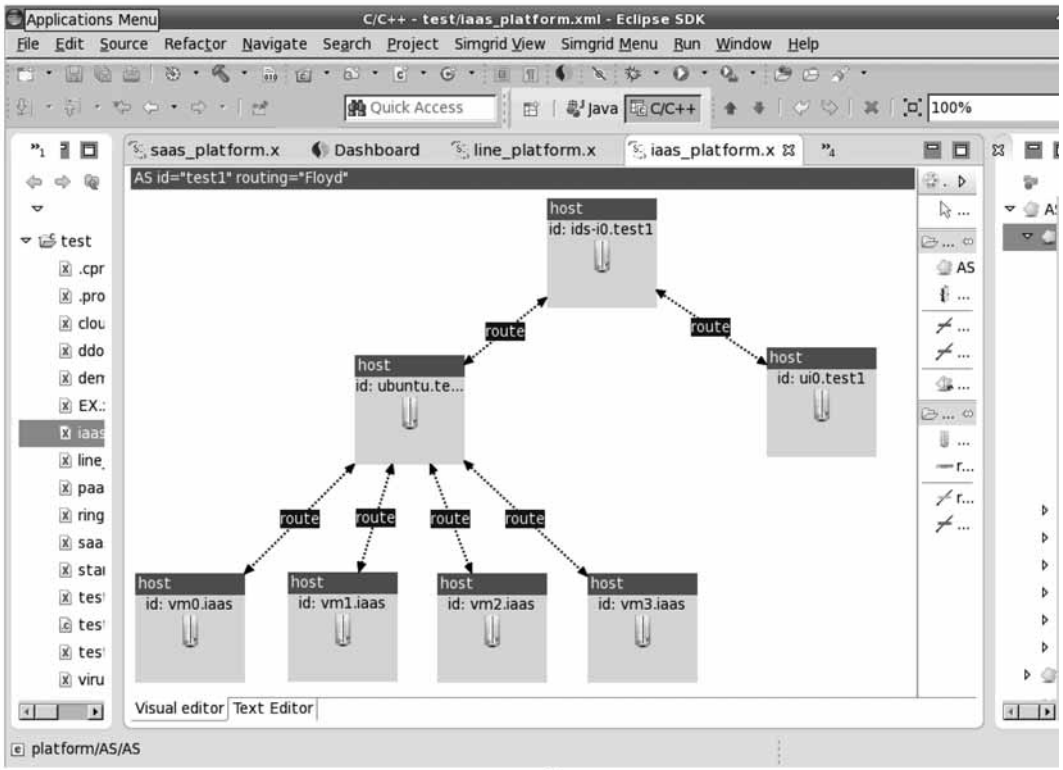
— по завершению выполнения сценария система развертывания осуществляет останов виртуальных машин и удаление временных файлов.

Входными данными для создания набора виртуальных машин являются общая конфигурация для виртуальных машин, конфигурация виртуальных сетей и набор шаблонов дисковых образов с предустановленными операционными системами и необходимым ПО. Конфигурационные файлы в данном случае имеют формат JSON и создаются редактором сценариев автоматически.

Используемый в системе сетевой сервис позволяет в режиме виртуального или гибридного моделирования выполнять на виртуальных машинах различные задания. Такой подход предоставляет возможность использовать в процессе моделирования реальное ПО, проводить более детализированные сценарии работы распределенной системы. Для использования какого-либо дополнительного ПО необходимо изменить исходные шаблоны дисковых образов.



а)



б)

Рис. 1. Интерфейс редактора конфигураций Eclipse/SimGrid:
а — пример задания системы с топологией "звезда"; *б* — пример задания системы с топологией "дерево"

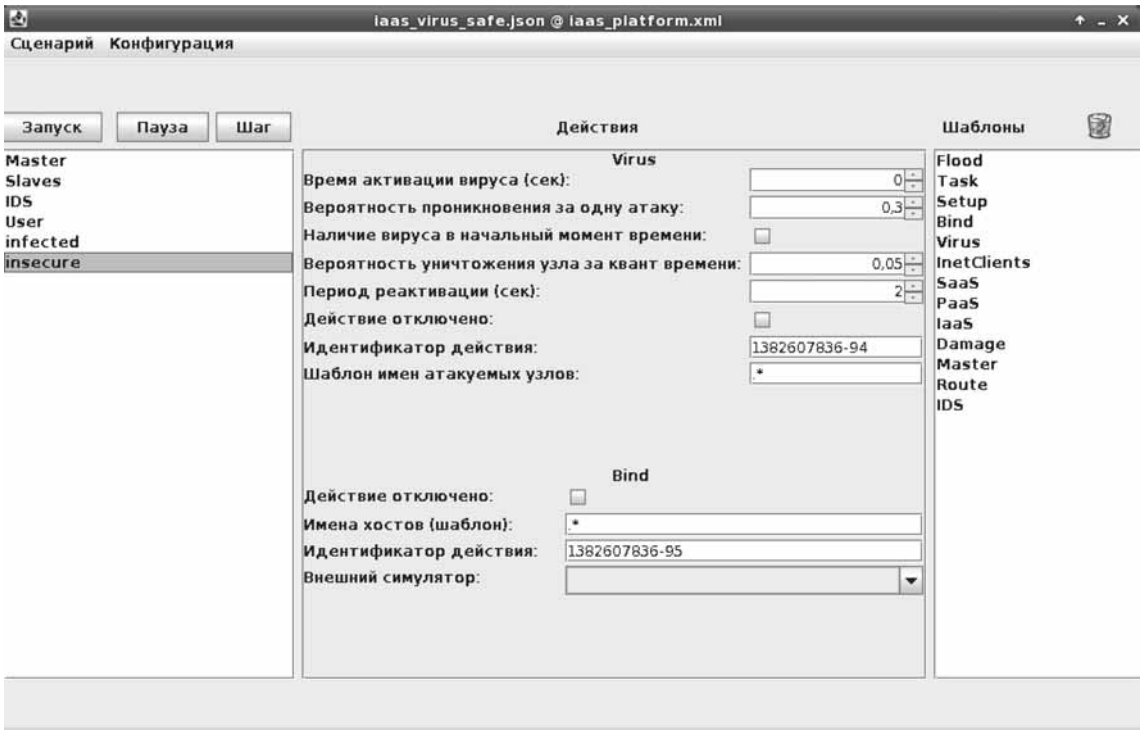


Рис. 2. Параметры активности в редакторе сценариев

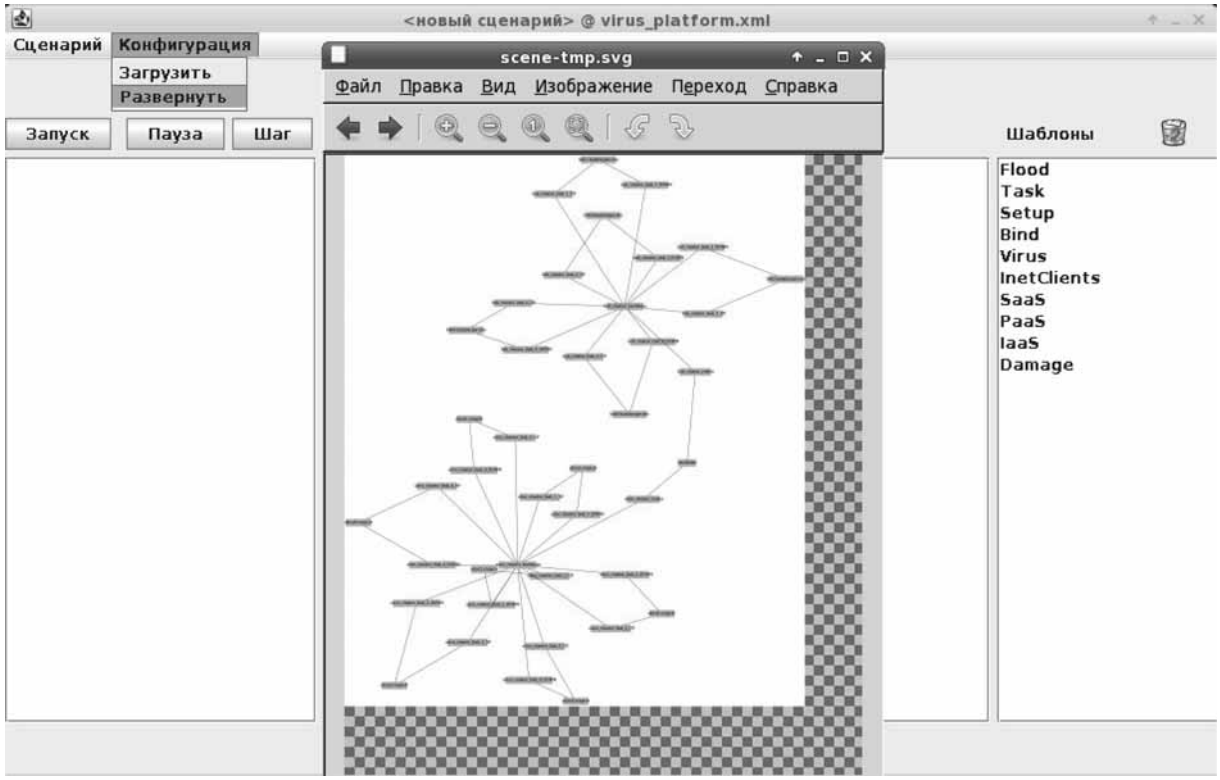


Рис. 3. Просмотр развернутой конфигурации

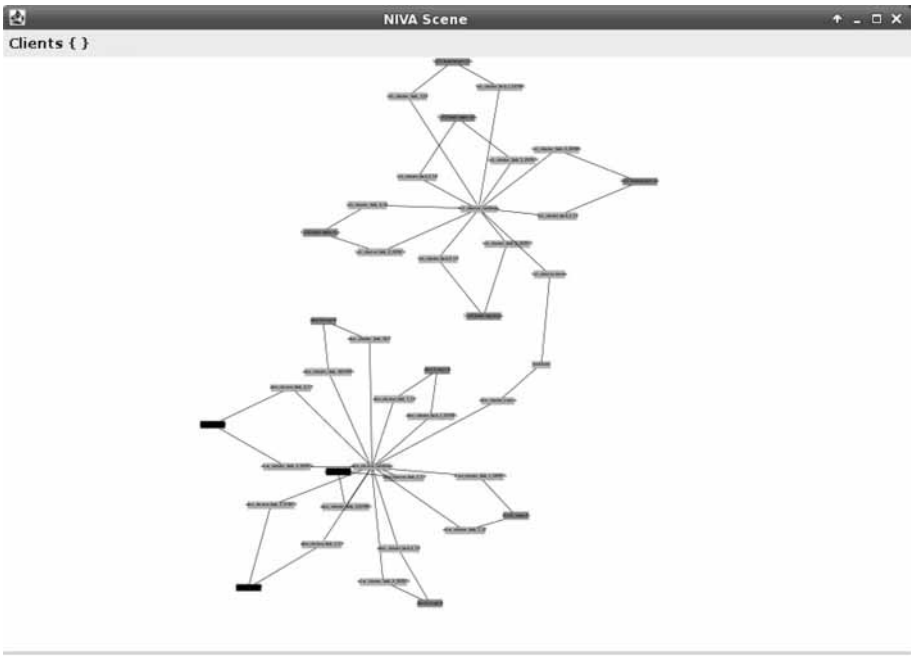


Рис. 4. Сервер визуализации "Сцена"

Использование созданного API

Далее описаны функциональные возможности системы НИВА и методы их расширения. Технологии, используемые при создании распределенных вычислительных систем, непрерывно эволюционируют вместе со средствами и методами их защиты. По этой причине среда моделирования Grid/Cloud-систем должна предусматривать добавление новых возможностей. Поскольку поведение системы при выбранной архитектуре неразрывно связано со сценарием ее использования, расширение функциональных возможностей должно выражаться в расширении следующих компонентов данного программного средства.

1. В меню графического редактора должны быть добавлены новые шаблоны действий, стадии работы которых (инициализация, реакции на приходящие сообщения и периодически выполняемые действия) параметризованы в достаточной степени, чтобы описать все тонкости нового поведения распределенной системы. Описание параметров действий реализуется на языке Java. Шаблон каждого действия при этом реализуется как простой класс вида JavaBeans (набор параметров с их значениями по умолчанию, текстовым описанием для подсказок оператору и с парой функций getter и setter). Никаких дополнительных функций от шаблонов действий не требуется, так как логика поведения действий программируется в динамически загружаемом модуле, относящемся к компоненту libscenario.

2. В библиотеку типовых шаблонов действий должны быть добавлены динамические подгружаемые модули, имеющие имена, соответствующие вновь добавленным шаблонам в редакторе. Такие модули на стадии инициализации способны принять установленные оператором или командным файлом параметры и функционировать в соответствии с их семантикой. При этом каждое действие может иметь свои, соответствующие его специфике переменные, в которых хранится его текущее состояние. Такое действие может также считывать и изменять переменные, отражающие состояния того узла, к поведению которого оно относится. Библиотека типовых шаблонов действий, как и базовый симулятор SimGrid, реализована на языке C. В распоряжении разработчика все возможности симулятора SimGrid, а также некоторые полезные функции-утилиты, упрощающие большинство типичных операций.

3. Если для повышения наглядности желательно каким-то новым способом отображать расширенное состояние вычислительных узлов, то можно расширить сценарий управления сервером визуализации "Сцена" (реализуется на языке JavaScript). На визуализатор "Сцена" могут выводить информацию также и другие симуляторы и процессы. Для этого можно воспользоваться библиотекой libscene.

4. В режиме гибридного моделирования, когда кроме базовой имитационной модели задействованы внешние симуляторы (например, на основе виртуальных машин), может потребоваться реализация сопряжения имитационной модели с реальными программами, запущенными во время моделирования на виртуальных либо на реальных компьютерах.

Добавление шаблона действий в редактор сценариев. Поскольку при расширении функций системы требуется добавление новых шаблонов действий, ниже рассматривается этот процесс и разбирается необходимая для этого последовательность действий. Каждому шаблону действий соответствует класс (и, соответственно, файл с расширением .java) пакета scenario.tmpl, причем все шаблоны действий должны наследовать базовый класс tmpl.Action. Кроме того, каждый класс данного типа должен следовать конвенции JavaBeans, т. е. иметь конструктор без параметров и публичные методы вида {getXXX, setXXX} для каждого параметра с именем XXX. Фактически последним требованием является использование аннотаций вида @UiLabel для того чтобы используемая библиотека MetaWidgets могла в наиболее удобной форме визуализировать параметры действий при их редактировании.

Примером такого класса может служить представленный далее шаблон действия типа Вирус.

```
package scenario.tmpl;

import scenario.*;
import org.metawidget.inspector.annotation.*;

@SuppressWarnings("serial")
public class Virus extends Action {
   double activ_t = 0.0;
   @UiLabel("Время активации вируса (сек)")
   public void setActiv_t(double x) { activ_t = x; }
   public double getActiv_t() { return activ_t; }
   double delta_t = 2.0;
   @UiLabel("Период реактивации (сек)")
   public void setDelta_t(double x) { delta_t = x; }
   public double getDelta_t() { return delta_t; }
   double affect_prob = 0.3;
   @UiLabel("Вероятность проникновения за одну атаку")
   public void setAffect_prob(double x) { affect_prob = x; }
   public double getAffect_prob() { return affect_prob; }
   double damage_prob = 0.03;
   @UiLabel("Вероятность уничтожения узла за квант времени")
   public void setDamage_prob(double x) { damage_prob = x; }
   public double getDamage_prob() { return damage_prob; }
   String targets = ".*";
   @UiLabel("Шаблон имен атакуемых узлов")
   public void setTargets(String x) { targets = x; }
   public String getTargets() { return targets; }
   boolean affected = false;
   @UiLabel("Наличие вируса в начальный момент времени")
   public void setAffected(boolean x) { affected = x; }
   public boolean getAffected() { return affected; }
   @Override
   public void apply(Host h) throws ValidEx {
   h.actions.add(this);
   }
}
```

После создания нового класса его нужно зарегистрировать в списке шаблонов, которые редактор предоставляет оператору во время редактирования (поле tmpl.Action.tmpl).

На этом модификации редактора сценариев заканчиваются. Вся остальная обработка происходит уже внутри компонента libscenario, ответственного за реализацию поведения действий во время моделирования.

Библиотека типовых шаблонов действий. Если для редактора сценариев, являющегося по сути разновидностью интегрированной среды разработки с графическим интерфейсом хорошо подходит язык программирования Java, то компонент libscenario, ответственный за интерпретацию действий во время моделирования, реализован на языке C. Это обеспечивает максимальную скорость работы, что особенно важно при многовариантном моделировании, а также упрощает интеграцию с базовым симулятором SimGrid.

После обработки и проверки корректности, определенной в редакторе сценариев конфигурации, создается временный файл в формате JSON, в котором для каждого узла распределенной системы установлен упорядоченный список действий, определяющих его поведение в процессе моделирования. Этот файл переводится при помощи автоматизированных средств, реализованных при помощи стандартного препроцессора, в бинарные данные, организованные в виде структур языка C, которые и параметризуют поведение логики шаблонов действий. При этом главная функция для выполнения действий не встраивается в библиотеку libscenario, а динамически загружается в виде DLL (файла с расширением .so), получая доступ к своим параметрам и внутреннему состоянию. Использование динамической загрузки повышает гибкость библиотеки. Возможен также переход на динамическую загрузку соответствующих классов и в редакторе сценариев.

Добавление шаблона действий в библиотеку типовых шаблонов. Для расширения библиотеки типовых шаблонов действий достаточно выполнить описанные ниже операции.

Для начала необходимо создать файл ИмяДействия.c, в начало и в конец которого вписать строки

```
#include "action_header.h"
[...]
#include "action_footer.h"
```

В этих специальных заголовочных файлах определены макросы, которые позволяют легко установить

входные параметры (они должны именоваться точно так же, как поля класса данного шаблона действий в графическом редакторе) и дополнительные переменные состояния, которые могут иметь произвольные типы. Эти макросы перечислены ниже.

Макрос	Что обозначает
P_LONG	Параметр типа long
P_DOUBLE	Параметр типа double
P_BOOL	Параметр типа bool(long)
P_STRING	Параметр типа const char*
DATA	Переменная состояния некоторого типа
QUEUE	Очередь структур некоторого типа

Кроме определения параметров, которые приходят из графического редактора, каждое действие может иметь дополнительные переменные состояния. Для их объявления можно использовать макросы DATA(type, name), а также QUEUE(type, queue). Если первый из них просто добавляет объявление соответствующей переменной в структуру состояния действия, то второй может быть удобен для организации очередей, что очень часто требуется при имитационном моделировании.

Макрос поддержки очереди объявляет два указателя — на первый элемент очереди и указатель на поле next последнего элемента. Очередь пуста тогда и только тогда, когда первый указатель равен NULL, при этом значение второго указателя может быть любым. Для работы с такой очередью подходит любая структура языка C, в которой имеется поле next соответствующего типа, а для добавления и извлечения элементов из очереди можно пользоваться макросами QPUT и QPOP соответственно.

В момент инициализации действия все дополнительные переменные (кроме параметров, которые устанавливаются в переданные из редактора значения) заполнены нулями, и функция действия init должна корректно установить начальные значения, если они должны быть отличны от нулей.

Для доступа к переменным состояния в функции действия передается указатель на структуру, в которой находятся все параметры и дополнительные переменные, поэтому доступ к ним из тела функций обычно выглядит как 'p->имяПараметраИлиПеременной'.

Рассмотрим в качестве первого примера логику простого вируса, который распространяется в Grid-среде посредством рассылки своего кода через уязвимости в сетевых сервисах.

```
#define PARAMS
   P_DOUBLE(activ_t)     \
   P_DOUBLE(delta_t)     \
   P_DOUBLE(affect_prob) \
   P_DOUBLE(damage_prob) \
   P_BOOL(affected)      \
   P_STRING(targets)

#include "action_header.h"

#define NAME "Virus"

static void init(params_t* p, name_t id) {
   markHostAffected(p->affected);
}

static task_action_t mesg(params_t *p, msg_task_t m, name_t tName) {
   if (strcmp(tName,NAME))
      return PASS;
   markHostAffected(p->affected |= occurred(p->affect_prob));
   return EXEC;
}
static void tick(params_t* p, double t) {
   if (!p->affected || t < p->activ_t)
      return;
   p->activ_t = t + randomize(p->delta_t);
   msg_task_t m = MSG_task_create(NAME, 1000, 1000, NULL);
   name_t target = selectRandomHost(p->targets);
   if (target == NULL)
      dmsg(stderr," %s: selectRandomHost(%s) failed\n", NAME, p->targets);
   else {
      MSG_task_isend(m, target);
      dmsg(stderr," %s sent: %s -> %s\n", NAME, hostName(), target);
   }
   markHostDamaged(occurred(p->damage_prob));
}
#include "action_footer.h"
```

В начале определяются параметры и дополнительные переменные состояния (PARAMS), которые в случае простой имитационной модели вируса имеют смысл, описанный ниже.

Параметр	Что определяет
activ_t	Время активации вируса
delta_t	Период реактивации
affect_prob	Вероятность заражения соседних узлов
damage_prob	Вероятность вывода из строя своего узла
affected	Заражен ли текущий узел
targets	Шаблон имен атакуемых узлов

Вся логика вируса, как и любого другого действия, состоит из функций init, mesg и tick, которые реализуют поведение в моменты инициализации, получения сообщения/задачи и периодическую активность в каждый квант времени (последняя функция вызывается периодически, однако это происходит не ранее, чем отработают все функции init данного узла).

Наиболее интересна в большинстве случаев функция mesg, которая принимает пришедшее сообщение в качестве аргумента. Поскольку на данном узле все действия вызываются последовательно, функции mesg всех действий вызываются от пришедшего сообщения до тех пор, пока одна из них не вернет статус обработки, отличный от PASS. Всего допустимо четыре следующих возможных реакции функции mesg на входящее сообщение.

Код	Семантика
PASS	Сообщение не относится к данному действию
EXEC	Сообщение обработано, нужно зарезервировать время на его обсчет
DROP	Сообщение обработано, выполнять связанные с ним вычисления не нужно
TRANSIT	Сообщение обработано для передачи дальше (считать ничего не нужно)

Как видно из представленного выше, только статус возврата PASS побуждает общую логику обработки пробовать применить дальнейшие действия к данному сообщению. Как только одно из действий вернет EXEC, DROP или TRANSIT, обработка считается завершенной, и статус обработки влияет лишь на дальнейшее время жизни и имитацию обсчета задачи/сообщения. Заметим, что если поведение узла моделируется во внешнем симуляторе, то выполнение задач проводится специфическим образом — они превращаются в реальные задачи и пересылаются на ту виртуальную машину либо тот реальный компьютер, где им положено вычисляться. Особенности поддержки гибридного моделирования рассмотрены выше в разд. "Гибридное моделирование".

Хотя наиболее важной характеристикой сообщений/задач являются их имена, с ними можно проассоциировать произвольные данные, которые хранятся в сообщении как тип void*.

В начале каждого кванта времени моделирования у действий вызывается функция tick. В имитационной модели время идет обычно гораздо быстрее, чем реальные "настенные часы". В используемом базовом симуляторе SimGrid время начинается с нуля и измеряется в секундах. Квант моделирования по умолчанию равен одной секунде.

Логика рассматриваемого вируса простая, что в данном случае не потребовало ввода дополнительных переменных для его состояния. Переход узла из нормального состояния в зараженное отмечается изменением параметра affected, начальное значение которого приходит из редактора сценариев. То есть параметры можно использовать также и как изменяемые переменные состояния.

В качестве более сложного примера можно рассмотреть имитацию работы уровня SaaS "эластичного облака":

```
#define PARAMS      \
   P_STRING(paas)   \
   P_LONG(highRqs)  \
   QUEUE(vm_t,vms)  \
   QUEUE(arq_t,arqs)

#include "action_header.h"
static void init(params_t* p, name_t id) {}

static task_action_t mesg(params_t *p, msg_task_t m, name_t tName) {
   vm_t* vm; arq_t* arq;
   task_action_t act = PASS;
   if (!strcmp(tName,"UTASK")) {
      arq = (arq_t*)MSG_task_get_data(m);
      assert(arq!=NULL && arq->vm==NULL && arq->task==m);
      QPUT(p->arqs,arq);
      act = TRANSIT;
   } else if (!strcmp(tName,"TAKE_VM")) {
      vm = (vm_t*)MSG_task_get_data(m);
      assert(vm!=NULL);
      dmsg("saas: got vm: %s\n",vm->name);
      QPUT(p->vms,vm);
      act = EXEC;
   }
   while (p->arqs!=NULL && p->vms!=NULL) {
      arq = QPOP(p->arqs);
      vm = QPOP(p->vms);
      MSG_task_isend(assignVM(arq,vm),vm->host);
```

```
    }
    return act;
  }
  static void tick(params_t* p, double t) {
    int n=0; arq_t* arq = p->arqs;
    while (n <= p->highRqs && arq!=NULL)
       { n++; arq = arq->next; }
    if (n >= p->highRqs)
       MSG_task_isend(MSG_task_create("ALLOC_PLATFORM",1,1,NULL),p->paas);
    else if (n==0 && p->vms!=NULL) {
       vm_t* vm = QPOP(p->vms);
       MSG_task_isend(MSG_task_create("FREE_VM",1,1,vm),vm->host);
    }
  }
```

В этом случае кроме входных параметров объявлены также две дополнительных очереди — очередь зарезервированных для расчетов виртуальных машин и очередь поступивших запросов. В случае как поступления нового запроса так и новой виртуальной машины (которые уровень SaaS запрашивает у уровня PaaS) поступившие данные просто заносятся в соответствующие очереди. Затем в цикле происходит ассоциирование имеющихся запросов с имеющимися в наличии виртуальными машинами и отсылка этих пар для обсчета.

Периодически вызываемая функция `tick` проверяет число скопившихся в очереди запросов и либо запрашивает дополнительные виртуальные машины, либо освобождает их в случае опустевшей очереди.

Расширение пространства состояний узлов. В некоторых случаях для того чтобы отразить логику работы различных действий на одном узле, может потребоваться расширение его пространства состояний. Переменные состояния описываются в файле host.d, где при помощи специальных макросов генерируются функции для обращения к состоянию узлов. Обращение через функции требуется для того чтобы изменения состояния автоматически визуализировались с помощью сервера "Сцена" и попадали в журнал в качестве значимых событий.

Программный интерфейс расширенного SimGrid. Ниже перечислены наиболее востребованные функции SimGrid и дополнительно имеющиеся в библиотеке libscenario, которыми можно пользоваться при программировании действий.

Функция	Описание
`MSG_task_get_data`	Получить ссылку на данные, ассоциированные с задачей
`MSG_task_isend`	Послать сообщение/задачу без ожидания факта приема
`MSG_task_create`	Создать сообщение/задачу с данным именем и ассоциированными данными
`selectRandomHost`	Выбрать случайный узел по шаблону имен
`hostName`	Имя текущего узла
`occurred`	Случилось ли событие с заданной вероятностью
`markHostAffected`	Пометить текущий узел как пораженный
`markHostDamaged`	Пометить текущий узел как выведенный из строя

Расширение возможностей сервера визуализации. Возможности сервера визуализации определяются сложностью загруженного в него изображения вместе с логикой встроенного в него JavaScript-кода. В простейшем случае этот сценарий имеет следующий вид:

```
function simgrid_onChange(host, color) {
document.getElementById(host).setAttribute
  ("fill", color);
}
```

В этом случае приходящие запросы всего лишь изменяют цвет изображенных узлов распределенной системы.

Использование libscene API. Общение между клиентом-симулятором и сервером "Сцена" происходит по единственному TCP-соединению. Имя сервера и порт должны быть доступны из переменной окружения SCENE в формате `host:port'. Если переменная SCENE не определена, режим интерактивной визуализации считается неактивным. Наличие переменной SCENE_DEBUG в окружении активирует отладочную печать сообщений протокола.

Функция инициализации `int scene_connect (const char *myNS, const char* svgURL, scene_event_handler_t evH)` возвращает следующие значения:

>0 — соединение установлено, возвращается версия протокола (например 1);

0 — режим интерактивной визуализации не активен (выключен);

<0 — ошибка.

Тип обработчика событий следующий:

```
typedef void (*scene_event_handler_t)(const
char* ns, const char* var, const char* value)
```

Функция `const char* scene_error()` расшифровывает возможные ошибки, например:

- не удалось обнаружить по указанному порту сервер "Сцена";
- клиент с таким именем уже присоединился и активен;
- передан `svgURL != NULL`, но предыдущий главный клиент (который установил SVG-картинку для визуализатора "Сцена") еще не отсоединился.

Функция `int scene_set(const char* ns, const char *var, const char *value)` посылает заявку на изменение переменной Scene. Возвращает 0 в случае успеха. Имя и величина не должны содержать символов типа кавычек и перевода строки.

Функция `int scene_process_events(ulong minEvs, ulong maxEvs)` и только она вызывает ус-

тановленный обработчик событий для еще не обработанных событий. Она обрабатывает не менее minEvs (возможно, дожидаясь недостающих), но не более maxEvs событий, после чего возвращает общее число обработанных событий. В режиме SCENE_DEBUG перед вызовами обработчика проводится соответствующая отладочная печать.

Заключение

В заключение приведем один пример использования системы. Для многовариантного моделирования 48 различных тестовых Grid/Cloud-конфигураций, потребовалось написать всего лишь следующие далее страницы кода на языке JavaScript:

```
// Scalar params
var run=true;
var save=true;
// Scalar/Array params
var prots=false;
var platforms='ring';
var attacks='virus';
// Update params from command line
args.forEach(function(a) { eval(String(a)); });
// Make arrays from scalars
if (!(prots instance of Array)) prots=[prots];
if (!(platforms instance of Array)) platforms=[platforms];
if (!(attacks instance of Array)) attacks=[attacks];
var ids, user, unsafe;
function set_cloud() {
   user = 'ui0.test1';
   ids = 'ids-i0.test1';
}
function set_grid() {
   user = 'ui0.user';
   ids = 'ids-re0.res';
}
platforms.forEach(function(platform) {
   eval('set_'+(platform.match(/.*aas/)?'cloud':'grid')+'();');
   attacks.forEach(function(attack) {
      prots.forEach(function(prot) {
         unsafe = !prot;
         log.add("\nplatform="+platform+", attack="+attack+", safe="+prot);
         eval('actions_'+platform+'();');
         actions_user();
         eval('actions_'+attack+'();');
         sc.gridConf('demos/multi/'+platform+'_platform.xml')
         if (save)
            sc.saveAs('/tmp/'+platform+'_'+attack+
                      '_'+(unsafe?'un':'')+'safe.json');
         if (run) {
              var res = JSON.parse(sc.run()); // JSON->object
              var total = res.ok + res.fail;
              if (total > 0)
                  res[" %ok"] = (100*res.ok)/total;
              log.add(res);
         }
         sc.closeQ();
      })
   })
});
function actions_user() {
   a = sc.activity('User');
   a.append('Bind', {'hosts':user});
   a.append('InetClients',{'servPat':ids,
                           'weight': 20000000000, 'bytes':500000,
                           'maxRqs':4,'rqTimeout':7,'sendProb':1});
}
function idsTaskRoute(to) {
   var a = sc.activity('IDS');
   a.append('Bind', {'hosts':ids});
   a.append('IDS', {'disabled':unsafe});
   a.append('Route',{'routeTo':to,'taskPat':'.*'});
```

```
}
function masterSlave(host,slaves) {
   var a = sc.activity('Master');
   a.append('Bind', {'hosts':host});
   a.append('Master',{'slavePat':slaves});
   a.append('IaaS', {'vmsOnHost':1});
   a = sc.activity('Slaves');
   a.append('Bind', {'hosts':slaves});
   a.append('IaaS', {'vmsOnHost':1});
}
function actions_iaas() {
   var iaas = 'ubuntu.test1';
   masterSlave(iaas,'vm.*iaas');
   idsTaskRoute(iaas);
}
function actions_paas() {
   var paas = 'broker.test1';
   masterSlave(paas,'node.*test1');
   idsTaskRoute(paas);
}
   function actions_saas() {
   var saas = 'broker.test1';
   masterSlave(saas,'node.*test1');
   idsTaskRoute(saas);
}
function actions_line() { actions_grid(); }
function actions_star() { actions_grid(); }
function actions_ring() { actions_grid(); }
function actions_grid() {
   var master = 'wms0.res';
   masterSlave(master,'wn.*\\.res');
   idsTaskRoute(master);
}
function actions_ddos_traffic() { actions_flood(2000000,1); }
function actions_ddos_process() { actions_flood(1,110000000000.0); }
function actions_flood(bytes,weight) {
   var a = sc.activity('DDoS');
   a.append('Bind', {'hosts':'eve[0-9]*\\.ev'});
   a.append('Flood',{'targets':ids,'bytes':bytes,'weight':weight,'freq':1});
}
function actions_virus() {
   var a = sc.activity('infected');
   a.append('Virus', {'affected':true,'targets':ids,'damage_prob':0});
   a.append('Bind', {'hosts':'.*\\.ev'});
   a = sc.activity('insecure');
   a.append('Virus', {});
   a.append('Bind', {'hosts':'.*'});
}
function actions_none() {}
```

Безусловно следует учитывать, что компактность кода в значительной степени объясняется выразительными возможностями самого языка JavaScript, который, с точки зрения пользователя, всего лишь расширен несколькими объектами и методами для доступа к разработанной среде моделирования НИВА. Полный прогон этого сценария в режиме с визуализацией, занимающий около часа, генерирует отчет, содержащий процент пораженных за время атаки узлов для каждого случая. Без визуализации программа отрабатывает все режимы за считанные секунды.

Список литературы

1. **SimGrid** — Versatile Simulation of Distributed Systems: Grids, Clouds. URL: http://simgrid.gforge.inria.fr/

2. **GridSim:** A Grid Simulation Toolkit for Resource Modelling and Application Scheduling for Parallel and Distributed Computing. URL: http://www.buyya.com/gridsim/

3. **CloudSim:** A Framework for Modeling and Simulation of Cloud Computing Infrastructures and Services. The Cloud Computing and Distributed Systems (CLOUDS) Laboratory, University of Melbourne. URL: http://www.cloudbus.org/cloudsim/

4. **Грушин Д. А., Поспелов А. И.** Система моделирования Grid: реализация и возможности применения // Труды Института системного программирования РАН. 2010. Т. 18. С. 243—260.

УДК 004.853

М. А. Нокель, аспирант, МГУ им. М. В. Ломоносова, e-mail: mnokel@gmail.com,
Н. В. Лукашевич, канд. физ.-мат. наук, вед. науч. сотр. НИВЦ МГУ им. М. В. Ломоносова

Тематические модели в задаче извлечения однословных терминов

Представлены результаты экспериментального исследования возможности использования тематических моделей в задаче автоматического извлечения однословных терминов. В качестве текстовых коллекций была взята подборка статей из электронных банковских журналов на русском языке и англоязычная часть корпуса параллельных текстов Europarl общественно-политической тематики. Эксперименты показывают, что использование тематической информации способно улучшить качество извлечения однословных терминов независимо от предметной области и целевого языка.

Ключевые слова: *тематические модели, кластеризация, извлечение однословных терминов*

M. A. Nokel, N. V. Loukachevitch

Topic Models in the Task of Single-Word Term Extraction

The paper describes the results of an experimental study of statistical topic models applied to the task of automatic single-word term extraction. The English part of the Europarl parallel corpus from the socio-political domain and the Russian articles taken from online banking magazines were used as target text collections. The experiments demonstrate that topic information can improve the quality of single-word term extraction regardless of the subject area and the target language.

Keywords: *topic models, clustering, single-word term extraction*

Введение

Извлечение терминов из текстов определенной предметной области играет значительную роль во многих задачах, в первую очередь — в разработке и пополнении различных терминологических ресурсов, таких как тезаурусы и онтологии [1]. Поскольку разработка таких ресурсов вручную достаточно трудоемка, за последние годы было проведено большое число исследований по автоматизации данного процесса.

Большинство современных методов извлечения терминов основывается на использовании различных статистических и лингвистических признаков слов. Основная цель при этом заключается в получении упорядоченного списка кандидатов в термины, в начале которого находится как можно больше слов, с наибольшей вероятностью являющихся терминами. В некоторых работах было экспериментально установлено, что использование машинного обучения для комбинирования признаков значительно улучшает результаты извлечения терминов по сравнению с методами, основанными только на одном каком-то признаке, поскольку те или иные признаки только частично отражают особенности поведения терминов в текстах [2].

На настоящее время традиционно используемые для извлечения терминов статистические признаки никак не отражают тот факт, что большинство терминов относятся к той или иной подтеме предметной области. Поэтому авторами было сделано предположение, что выделение таких подтем в коллекции текстов способно улучшить качество автоматического извлечения терминов. Для проверки этого предположения будут рассмотрены различные методы выделения подтем, которые часто в литературе называют статистическими тематическими моделями [3].

Некоторые виды статистических тематических моделей могут основываться на традиционных методах автоматической кластеризации текстов [4]. В последнее время предложены вероятностные механизмы выделения подтем в текстовых коллекциях, такие как методы, основанные на скрытом распределении Дирихле (Latent Dirichlet allocation [3]), которые собственно и были названы статистическими тематическими моделями и в настоящее время интенсивно исследуются в рамках различных приложений автоматической обработки текстов ([4—6]).

Основная задача работы, описанной в данной статье, заключается в исследовании возможности использования тематической информации для повышения качества извлечения однословных терминов. Для этой цели вначале в текстовой коллекции выделяют подтемы, затем к ним применяют некоторые модификации хорошо известных признаков, которые впоследствии используют вместе с другими статистическими и лингвистическими признаками.

Для того чтобы результаты, представленные в статье, не зависели ни от предметной области, ни от языка, были взяты две текстовые коллекции двух различных предметных областей: тексты банковской предметной области на русском языке и речи с заседаний Европарламента общественно-политической предметной области на английском языке. При этом эксперименты построены следующим образом: вначале статистические тематические модели исследованы с точки зрения задачи извлечения терминов в целях выбора лучшей; затем осуществлено сравнение признаков, посчитанных для лучшей тематической модели, с другими признаками для изучения вклада, который дает использование тематической модели.

Близкие работы

За последние годы было предложено много различных статистических и лингвистических признаков слов, используемых для извлечения однословных терминов из коллекции текстов определенной предметной области ([7—10] и др.).

Можно выделить несколько групп существующих признаков.

- *Признаки, основанные на частотности слов-кандидатов.* К этой группе относится, например, признак TFRIDF, предложенный в работе [7] и использующий модель Пуассона для предсказания терминологичности слов.
- *Признаки, использующие контрастную коллекцию*, т. е. коллекцию более общей тематики. Одним из наиболее характерных представителей данной группы является широко используемый на практике признак *относительная частотность* [8], основанный на сравнении относительных частотностей слов в рассматриваемой и в контрастной текстовой коллекциях.
- *Контекстные признаки*, соединяющие в себе информацию о частотности слов-кандидатов с данными о контексте их употребления. Наиболее известными признаками в этой группе являются C-Value [9] и NC-Value [10], учитывающие частоту встречаемости объемлющего словосочетания для слова-кандидата.

Однако ни один из предложенных признаков не является определяющим [11], и фактически из текстов извлекается довольно большой список слов-кандидатов, которые затем должны быть проанализированы и подтверждены экспертом по предметной области. По этой причине важно дополнять список используемых признаков, что позволит получать в начале списка как можно больше слов, с наибольшей вероятностью являющихся терминами. В данной статье введен качественно новый тип признаков, основывающихся на использовании тематической информации.

Отдельно следует отметить работу [12], в которой решалась задача выявления терминов, наиболее характеризующих рубрики, к которым относится документ. Однако в исследуемом подходе разделение документов на рубрики неизвестно, и для определения подтем в коллекции применяется представленный далее математический аппарат.

Статистические тематические модели

Новые признаки слов-кандидатов, которые вводятся в данной статье, используют информацию, получаемую статистическими тематическими моделями в исследуемых текстовых коллекциях.

Статистическая тематическая модель (далее — тематическая модель) коллекции текстовых документов на основе статистических методов определяет, к каким подтемам относится каждый документ и какие слова образуют каждую подтему, представляющую собой список часто встречающихся рядом друг с другом слов, упорядоченный по убыванию степени принадлежности ему [13]. Так, в табл. 1 представлены первые пять слов, наиболее полно характеризующие три случайно выбранных подтемы, выделенные из русскоязычных текстов банковской тематики рассматриваемой коллекции.

Таблица 1

Примеры подтем

Подтема 1	Подтема 2	Подтема 3
Банкнота	*Обучение*	*Германия*
Оффшорный	*Студент*	*Франция*
Счетчик	*Учебный*	*Евро*
Купюра	*Вуз*	*Европейский*
Подделка	*Семинар*	*Польша*

В тематических моделях, как правило, используется модель мешка слов, в которой каждый документ рассматривается как набор встречающихся в нем слов. При этом перед выделением подтем текстовая коллекция обычно подвергается предобработке, выделяющей только значимые слова в каждом документе. В частности, в данном исследовании для русского языка были отобраны только существительные и прилагательные, а для английского — только существительные, поскольку они покрывают большую часть терминов.

На настоящее время разработано достаточно много различных тематических моделей. В целях выбора моделей для исследования были проанализированы предыдущие работы, в которых осуществлялось сравнение моделей с точки зрения различных практических приложений. Так, в работе [5] утверждается, что каждая тематическая модель имеет свои сильные и слабые стороны. Сравнивая между собой методы NMF (неотрицательной матричной факторизации) и LDA (латентного размещения Дирихле), авторы приходят к выводу, что оба этих алгоритма дают похожее качество, хотя NMF и выдает немного больше бессвязных подтем. В работе [6] утверждается, что традиционные тематические модели показывают приемлемое качество выделения подтем, но имеют множество ограничений. В частности, они предполагают, что каждый документ имеет только одну тематику. В действительности же документы представляют собой, как правило, смесь подтем. Кроме того, авторы отмечают, что параметры традиционных моделей достаточно сложно настраивать. В то же время в работе подчеркивается, что более сложные модели (такие как LDA) необязательно дадут лучшие результаты.

Поскольку, как следует из упомянутых выше работ, среди тематических моделей нет явного лидера и непо-

нятно, какое качество они покажут в задаче извлечения однословных терминов, было решено выбрать несколько наиболее характерных представителей, которых можно отнести либо к вероятностным методам, либо к методам кластеризации текстов, рассматриваемых с точки зрения тематических моделей. Каждая из выбранных моделей будет рассмотрена в следующих подразделах.

Тематические модели, основанные на кластеризации текстов

Традиционные тематические модели, как правило, основываются на методах жесткой кластеризации, рассматривающих каждый документ как разреженный вектор в пространстве слов большой размерности [14]. После окончания работы алгоритма кластеризации каждый получившийся кластер рассматривается как один большой документ для вычисления вероятностей входящих в него слов по следующей формуле:

$$P(w|t) = \frac{TF(w|t)}{\sum\limits_{w} TF(w|t)},$$

где $TF(w|t)$ — частотность слова w в кластере t.

В данной статье описано исследование перечисленных далее широко известных алгоритмов кластеризации.

♦ **Алгоритм *k*-средних и сферический алгоритм *k*-средних.** Алгоритм *k*-средних [15] начинает свою работу со случайной инициализации центров масс каждого кластера. Далее он итеративно повторяет следующие шаги:

- все документы разбиваются на кластеры в соответствии с тем, какой из центров масс оказался ближе по выбранной метрике;
- для каждого кластера пересчитывается центр масс.

В качестве метрики близости между двумя документами исследовались следующие метрики (здесь и дальше через **A** и **B** обозначены два произвольных вектора, представляющие собой документы).

- Евклидово расстояние (*K-Means*) [15]:

$$\mathrm{sim}(\mathbf{A}, \mathbf{B}) = \sqrt{\sum_i (A_i - B_i)^2}.$$

- Косинусная мера близости (*сферический алгоритм k-средних — SPK-Means*). При этом все векторы, представляющие документы, нормализуются к единичной гиперсфере [16]:

$$\mathrm{sim}(\mathbf{A}, \mathbf{B}) = \frac{\sum\limits_i (A_i \times B_i)}{\sqrt{\sum\limits_i A_i} \times \sqrt{\sum\limits_i B_i}}.$$

♦ **Иерархическая агломеративная кластеризация**. Данный алгоритм [17] вначале рассматривает каждый документ как отдельный кластер. Затем он итеративно, пока не останется заданное число кластеров, повторяет следующие шаги:

- находятся и объединяются в кластер два наиболее близких кластера;
- вычисляются расстояния между новым кластером и другими.

В качестве способов определения наиболее близких кластеров исследовались следующие наиболее распространенные способы [17]:

- *complete-link* (полное связывание); наиболее близкими считаются кластеры с наименьшим максимальным парным расстоянием между документами;
- *single-link* (одиночное связывание); наиболее близкими считаются кластеры с наименьшим минимальным парным расстоянием между документами;
- *average-link* (среднее связывание); наиболее близкими считаются кластеры с наименьшим средним парным расстоянием между документами.

♦ **Метод неотрицательной матричной факторизации (NMF),** изначально разработанный для уменьшения размерности и использующийся для решения задач кластеризации [18]. Данный алгоритм осуществляет нечеткую кластеризацию, относящую один и тот же документ к разным кластерам с разными вероятностями.

Принимая на входе неотрицательную разреженную матрицу **V**, которая получается записыванием векторов, представляющих документы, по столбцам, алгоритм ищет такие матрицы **W** и **H** меньшей размерности, что $\mathbf{V} \approx \mathbf{WH}$ по некоторой метрике. В качестве такой метрики исследовались следующие [19]:

- евклидово расстояние (*NMF Euc*):

$$\|\mathbf{A} - \mathbf{B}\|^2 = \sum_{i,j} (A_{i,j} - B_{i,j})^2;$$

- расстояние Кульбака-Лейблера (*NMF KL*):

$$D(\mathbf{A}\|\mathbf{B}) = \sum_{i,j} \left(A_{i,j} \log \frac{A_{i,j}}{B_{i,j}} - A_{i,j} + B_{i,j}\right).$$

В результате работы алгоритма в матрице **W** получается распределение слов по кластерам, а в матрице **H** — распределение документов по кластерам. Нормируя соответствующие величины для каждого слова/документа, можно получить вероятности принадлежности этого слова/документа кластеру.

Вероятностные тематические модели

Вероятностные тематические модели представляют каждый документ в виде смеси подтем, в которой каждая подтема представляет собой некоторое вероятностное распределение над словами. Вероятностные модели порождают слова по правилу

$$P(w|d) = \sum_t P(w|t)P(t|d),$$

где $P(t|d)$ и $P(w|t)$ — распределение подтем по документам и слов по подтемам, а $P(w|d)$ — наблюдаемое распределение слов по документам. Порождение происходит следующим образом. Для каждого текста d и для каждого слова $w \in d$ выбирается тема t из распределения $P(t|d)$ и затем генерируется слово w из распределения $P(w|t)$.

В данной работе рассматриваются следующие наиболее известные методы построения вероятностных тематических моделей.

1. Метод вероятностного латентного семантического индексирования (PLSI), предложенный в работе [20]. Данный метод моделирует матрицу **V** ($V_{i,j}$ обозначает

число вхождений слова w_i в документ d_j), получающуюся из модели с k подтемами:

$$P(w_i, d_j) = \sum_{t=1}^{k} P(t)P(d_j|t)P(w_i|t).$$

Поскольку в статье [21] теоретически обосновано, что алгоритм NMF, минимизирующий расстояние Кульбака-Лейблера и рассмотренный в предыдущем разделе, эквивалентен алгоритму PLSA, в данном исследовании метод PLSA не рассматривается отдельно.

2. Латентное размещение Дирихле (LDA), предложенное в работе [3]. Модель LDA расширяет модель PLSI, добавляя туда априорное распределение параметров модели ($P(w|t)$ и $P(t|d)$), считая их распределенными по закону Дирихле. Для настройки параметров модели необходим Байесовский вывод. Однако поскольку он алгоритмически неразрешим [3], в работе исследовались следующие два приближенных способа Байесовского вывода:

- LDA VB — вариационный Байесовский вывод [3];
- LDA Gibbs — метод Монте-Карло с марковскими цепями, использующий сэмплирование Гиббса [22].

Базовая "тематическая" модель

В качестве baseline была взята "тематическая" модель, которая не выделяет никаких подтем, а просто рассматривает каждый документ как отдельно взятую подтему. Данная модель использовалась в экспериментах для сравнения с другими методами.

Коллекции текстов для экспериментов

Во всех экспериментах, которые описаны в данной статье, слова-кандидаты извлекались из двух перечисленных далее коллекций.

- Коллекция банковских русскоязычных текстов (10422 документа, примерно 15,5 млн слов), взятых из различных электронных банковских журналов, включая "Аудитор", "Банки и Технологии", "РБК" и др.;
- Английская часть корпуса параллельных текстов Europarl [23] из заседаний Европарламента (9673 документа, примерно 54 млн слов).

Для подтверждения терминологичности слов-кандидатов использовались следующие "золотые стандарты".

- Для русского языка — тезаурус, разработанный вручную для Центрального банка Российской Федерации и включающий в себя порядка 15000 терминов, относящихся к сфере банковской активности, денежной политики и макроэкономики.
- Для английского языка — официальный многопрофильный тезаурус Европейского Союза Eurovoc [24], предназначенный для ручного индексирования заседаний Европарламента. Его английская версия включает в себя 15161 термин.

При этом слово-кандидат считается термином, если оно содержится в тезаурусе. В качестве метрики оценки качества была выбрана средняя точность (*AvP*) [16], которая определяется для множества D всех слов-кандидатов и его подмножества $D_q \subseteq D$, представляющего действительно термины (т. е. подтвержденные тезаурусом) как

$$AvP(n) = \frac{1}{|D_q|} \sum_{1 \leqslant k \leqslant |D_q|} \left(r_k \left(\frac{1}{k} \sum_{1 \leqslant i \leqslant k} r_i \right) \right),$$

где $r_i = 1$, если i-е слово-кандидат $\in D_q$, и $r_i = 0$ иначе. Данная формула отражает тот факт, что чем больше терминов сосредоточено в вершине итогового списка слов-кандидатов, тем выше мера средней точности. Поскольку все признаки слов-кандидатов рассчитывались для 5000 самых частотных слов, далее в статье будет использоваться мера средней точности на данном уровне — *AvP*@5000.

Эксперименты проводились с разным числом выделяемых подтем, а именно 50, 100 и 150. Визуально результаты получались разными, однако на качестве извлечения терминов это никак не отразилось. По этой причине все дальнейшие эксперименты проводились с числом подтем, равным 100.

Выбор лучшей тематической модели

Как уже было отмечено ранее, вначале представлены результаты экспериментов по определению наилучшей тематической модели. С этой целью предложены и рассчитаны для каждой из рассмотренных выше тематических моделей некоторые модификации известных признаков слов.

Признаки, использующие тематическую информацию

Основной идеей всех признаков, использующих полученную с помощью какой-либо тематической модели информацию, является тот факт, что в начале списков, образующих подтемы, с большой вероятностью находятся термины. Для экспериментов были предложены некоторые модификации известных признаков (табл. 2). В табл. 2 используются следующие обозначения:

- $TF(w)$ — частотность слова w;
- $DF(w)$ — документная частотность слова w;
- $P(w|t)$ — условная вероятность принадлежности слова w подтеме t;
- k — число топиков.

Таблица 2

Признаки, использующие тематическую информацию

Признак	Формула
Частотность (TF)	$\sum_t P(w\|t)$
TFIDF	$TF(w)\log\frac{k}{DF(w)}$
Domain Consensus (DC) [25]	$-\sum_t (P(w\|t)\log P(w\|t))$
Maximum TF (MTF)	$\max_t P(w\|t)$
Term Score (TS) [6]	$\sum_t TS(w\|t)$, где $TS(w\|t)\log\frac{P(w\|t)}{(\prod_t P(w\|t))^{\frac{1}{k}}}$
TS-IDF	$TS(w)\log\frac{k}{DF(w)}$
Maximum TS (MTS)	$\max_t TS(w\|t)$

Таблица 3

Средняя точность *AvP*@5000 признаков в русском корпусе

Модель	Признак						
	TF	TFIDF	DC	MTF	TS	TSIDF	MTS
K-Means	33,3	25,5	32,7	34,4	35,7	28,7	34,3
SPK-Means	35,5	27,2	35	33,9	36,3	30,1	33,6
Single-link	34,8	39,9	33,6	38,9	38,4	40,5	39
Complete-link	35,6	41	34,5	39,2	38,4	41	39,5
Average-link	35,8	40,7	34,5	39,5	39	40,9	39,6
NMF Euc	40,8	42,5	40,3	40,8	42	43,1	41,9
NMF KL	42,3	40,3	37,5	47,1	**48,9**	42,9	47,9
LDA VB	35,8	42,7	32,8	42,8	42,5	45,1	46,5
LDA Gibbs	37,7	38,4	35	46,2	42,6	42,8	47,2
Baseline	34	37,6	32,8	38,5	38,1	42	38,1

Таблица 4

Средняя точность *AvP*@5000 признаков в английском корпусе

Модель	Признак						
	TF	TFIDF	DC	MTF	TS	TSIDF	MTS
K-Means	29,3	32,3	28,9	30,3	30,1	31,8	30,4
SPK-Means	28,1	29,8	27,9	28,7	28,6	29,7	28,7
Single-link	30,3	38,9	29,8	37,3	36,5	38,8	39,9
Complete-link	31,1	39,6	30,4	37,2	34,6	38,9	39
Average-link	30,5	38,9	29,9	37,1	35,4	38,3	39,3
NMF Euc	34,4	31,6	32,3	41,1	43,7	31,6	40,5
NMF KL	33,3	37,7	31,2	44,3	**44,4**	37,3	44,1
LDA VB	32,3	30,3	30,5	37,1	36,3	30,3	38,5
LDA Gibbs	35,2	41,8	33,3	42,6	37,8	43,7	43,5
Baseline	31,5	32,8	30	36	33,6	35	36,7

Результаты экспериментов

В табл. 3 и 4 представлены результаты экспериментов для исследуемых русского и английского корпусов соответственно (средние точности лучших признаков выделены полужирным шрифтом).

Как видно из представленных таблиц, лучшее качество независимо от языка и предметной области дает тематическая модель NMF, минимизирующая расстояние Кульбака-Лейблера. Так, лучшим признаком для обоих языков является *Term Score* с $\frac{48,9-42}{42}\times 100\ \% \approx 16\ \%$ (соответственно $\frac{44,4-36,7}{36,7}\times 100\ \% \approx 21\ \%$) прироста качества относительно лучших признаков базовой модели.

Кроме вычисления средней точности отдельных признаков было также осуществлено их комбинирование для каждой исследуемой тематической модели в отдельности с помощью метода логистической регрессии, реализованного в библиотеке Weka [26]. При этом проводилась четырехкратная кросс-проверка, означающая, что вся исходная выборка разбивалась случайным образом на четыре равные непересекающиеся части. Каждая часть при этом по очереди становилась контрольной подвыборкой, а обучение проводилось по остальным трем. Результаты комбинирования признаков для русского и английского корпусов представлены в табл. 5.

Как следует из представленных выше таблиц, тематическая модель NMF, минимизирующая расстояние Кульбака-Лейблера, снова дает наилучшее качество с 10 % прироста для русского и с 23 % прироста для английского корпусов относительно базовой тематической модели. Таким образом, наилучшей тематической моделью оказалась модель NMF, минимизирующая расстояние Кульбака-Лейблера.

Таблица 5

Средняя точность *AvP*@5000 комбинирования признаков, использующих тематическую информацию

Модель	Для русского корпуса	Для английского корпуса
Baseline	44,9	36,2
K-Means	36,2	33,7
SPK-Means	38,1	33,3
Single-link	42,1	41,4
Complete-link	41,9	41,3
Average-link	42,7	41,3
NMF Euc	43,4	43,8
NMF KL	**49,5**	**44,5**
LDA VB	46,1	36,7
LDA Gibbs	47,9	44,4

Сравнение с другими признаками

Для изучения вклада тематической информации в задачу извлечения однословных терминов было решено сравнить результаты предложенных признаков, использующих тематическую информацию, с остальными статистическими и лингвистическими признаками для обоих исследуемых корпусов для 5000 самых частотных слов.

В качестве признаков, не использующих тематическую информацию, были взяты характерные представители групп, описанных в разделе "Близкие работы".

Признаки, основанные на частотности

Признаки из данной группы опираются на предположение о том, что термины, как правило, встречаются в коллекции гораздо чаще остальных слов. В исследование были включены следующие признаки: *частотность*; *документная частотность*; *TFIDF* [27]; *TFRIDF* [7]; *domain Consensus* [25].

Признаки, использующие контрастную коллекцию

Для вычисления признаков этой категории кроме целевой коллекции текстов предметной области использовалась контрастная коллекция текстов более общей тематики. Для русского языка в качестве таковой была взята подборка из примерно одного миллиона новостных текстов, а для английского — *n*-граммные статистики из Британского Национального Корпуса [28].

Основная идея таких признаков заключается в том, что частотности терминов в целевой и контрастной коллекциях существенно различаются. В данном исследовании рассматривались следующие признаки: относительная частотность [8]; релевантность [29]; TFIDF [27] с вычислением документной частотности по контрастной коллекции; Contrastive Weight [30]; Discriminative Weight [31]; KF-IDF [32]; Lexical Cohesion [33]; логарифм правдоподобия [34].

Контекстные признаки

Контекстные признаки соединяют в себе информацию о частотности слов-кандидатов с данными о контексте их употребления в коллекции. В данном исследовании рассматривались следующие признаки: C-Value [9]; NC-Value; MNC-Value [10]; Token-LR; Token-FLR; Type-LR; Type-FLR [35]; Sum3; Sum10; Sum50; Insideness [2].

Прочие признаки

В качестве остальных признаков, не использующих тематическую информацию, рассматривались: номер позиции первого вхождения в документы; типы слов-кандидатов (существительное или прилагательное); слова-кандидаты, начинающиеся с заглавной буквы; существительные в именительном падеже (подлежащие) и слова из контекстного окна с некоторыми самыми частотными предопределенными терминами [36].

Кроме перечисленных выше признаков рассматривались также и комбинации данных признаков с некоторыми статистическими величинами (такими как частотность в целевом корпусе). Всего было взято 28 таких признаков.

Результаты экспериментов

Лучшие признаки каждой из упомянутых выше групп для русского и английского корпусов приведены в табл. 6 и 7.

Как видно из приведенных таблиц, независимо от языка и предметной области, лучшими индивидуальными признаками оказались тематические, превзойдя остальные на 19 и 15 % средней точности для русского и английского корпусов соответственно.

Для оценки же вклада тематических признаков в общую модель извлечения однословных терминов сравнили модель извлечения, учитывающую тематические признаки (7 baseline-признаков и 7 признаков, посчитанных для наилучшей тематической модели NMF KL), и модель, не использующую их. Результаты сравнения для обоих рассматриваемых корпусов представлены в табл. 8 (комбинирование признаков осуществлялось с помощью логистической регрессии из библиотеки Weka [27]).

По мнению авторов, данные результаты, показанные на двух разных коллекциях, подтверждают, что тематические модели действительно вносят дополнительную информацию в процесс извлечения терминов.

В заключение представлены первые десять элементов из списков извлеченных слов-кандидатов, полученных с помощью моделей, учитывающих тематические признаки (термины выделены курсивом).

- Для русского корпуса: *Банковский*, *Банк*, Год, *РФ*, *Кредитный*, *Налоговый*, *Кредит*, *Пенсионный*, Средство, Клиент;

Таблица 6

Средняя точность признаков в русском корпусе

Группа признаков	Лучший признак	*AvP*@5000
Основанные на частотности	TFRIDF	41,1
Использующие контрастный корпус	Логарифм правдоподобия	36,9
Контекстные	Sum3	37,4
Тематические	Term Score	**48,9**

Таблица 7

Средняя точность признаков в английском корпусе

Группа признаков	Лучший признак	AvP@5000
Основанные на частотности	TFRIDF с подлежащими	38,5
Использующие контрастный корпус	TFIDF с подлежащими	34,2
Контекстные	C-Value	31,3
Тематические	Term Score	**44,5**

Таблица 8

Средняя точность *AvP*@5000 моделей с тематическими признаками и без них

Корпус	Без тематических признаков	С тематическими признаками
Русский	54,6	**56,3**
Английский	50,4	**51,4**

• Для английского корпуса: Member, Minute, *Amendment*, *Document*, *EU*, *President*, People, *Directive*, Year, Question.

Заключение

В статье представлены результаты экспериментального исследования возможности применения тематических моделей для улучшения качества автоматического извлечения однословных терминов.

Исследованы различные тематические модели (как вероятностные, так и традиционные методы кластеризации) и предложено несколько модификаций известных признаков для упорядочивания слов-кандидатов по убыванию их терминологичности. В качестве текстовых коллекций были взяты два различных корпуса: электронные банковские статьи на русском языке и корпус Europarl на английском языке.

Эксперименты показали, что независимо от предметной области и языка использование тематической информации способно значительно улучшить качество автоматического извлечения однословных терминов.

Список литературы

1. **Лукашевич Н. В.** Тезаурусы в задачах информационного поиска. М.: Изд-во Московского университета, 2011.

2. **Loukachevitch N.** Automatic Term Recognition Needs Multiple Evidence // In the Proceedings of the 8th International Conference on LREC, Istanbul, Turkey. May 21—27, 2012. URL: http://www.lrec-conf.org/proceedings/lrec2012/pdf/893_Paper.pdf.

3. **Blei D., Ng A. and Jordan M.** Latent Dirichlet Allocation // Journal of Machine Learning Research. 2003. № 3. P. 993—1022.

4. **He Q., Chang K., Lim E., Banerjee A.** Keep It Smile with Time: A Reeximanation of Probabilistic Topic Detection Models // In the Proceedings of IEEE Transactions Pattern Analysis and Machine Intelligence 2010. IEEE Computer Society, 2010. Vol. 32, Is. 10. P. 1795—1808.

5. **Stevens K., Kegelmeyer P., Andrzejewski D., Buttler D.** Exploring Topic Coherence over many models and many topics // In the Proceedings of EMNLP-CoNLL. Jeju Island, Korea. July 12—14, 2012. Association for Computational Linguistics. 2012. P. 952—961.

6. **Blei D. and Lafferty J.** Topic Models // Text Mining: Classification, Clustering and Applications. Chapman & Hall, 2009. P. 71—89.

7. **Church K. and Gale W.** Inverse Document Frequency IDF. A Measure of Deviation from Poisson // In the Proceedings of the Third Workshop on Very Large Corpora. Cambridge, MA. June 30, 1995. MIT Press, 1995. P. 121—130.

8. **Ahmad K., Gillam L., Tostevin L.** University of Survey Participation in Trec8. Weirdness indexing for logical document extrapolation and retrieval // In the Proceedings of TREC 1999. Gaithersburg, Maryland, November 17—19, 1999. URL: http://trec.nist.gov/pubs/trec8/papers/surrey2.pdf

9. **Nakagawa H. and Mori T.** A Simple but Powerful Automatic Term Extraction Method // In the Proceedings of the Second International Workshop on Computational Terminology. 2002. Association for Computational Linguistics, 2002. P. 29—35.

10. **Frantzi K. and Ananiadou S.** Automatic Term Recognition Using Contextual Cues // In the Proceedings of the IJCAI Workshop on Computational Terminology. Taipei, Taiwan, 2002. Association for Computational Linguistics, 2002. P. 29—35.

11. **Pecina P. and Schlesinger P.** Combining Association Measures for Collocation Extraction // In the Proceedings of the COLING/ACL. Sydney, Australia. July 17—21, 2006. ACL Press, 2006. P. 651—658.

12. **Голомазов Д. Д.** Выделение терминов из коллекции текстов с заданным тематическим делением // Информационные технологии. 2010. № 2. С. 8—13.

13. **Воронцов К. В., Потапенко А. А.** Регуляризация, робастность и разреженность вероятностных тематических моделей // Компьютерные исследования и моделирование. 2012. Т. 4, № 12. С. 693—706.

14. **Salton G.** Automatic text processing: the transformation, analysis, and retrieval of information by computer. Addison-Wesley Longman Publishing Co., 1989.

15. **Kurz D. and Xu F.** Text Mining for the Extraction of Domain Retrieval Terms and Term Collocations // In the Proceedings of the International Workshop on Computational Approaches to Collocations, Vienna, Austria. July 22—23, 2002. URL: http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.86.7929&rep=rep1&type=pdf

16. **Zhong Shi.** Efficient Online Spherical K-means Clustering // In the Proceedings of IEEE-IJCNN, Montreal, Canada. July 31 — August 4, 2005. P. 3180—3185. URL: http://www.sci.utah.edu/~ weiliu/research/clustering_fmri/Zhong_sphericalKmeans.pdf

17. **Johnson S. C.** Hierarchical Clustering Schemes // Psychometrika 1967. № 2. P. 241—254.

18. **Xu W., Liu X., Gong Y.** Document Clustering Based On Non-negative Matrix Factorization // In the Proceedings of SIGIR, Toronto, Canada. July 28 — August 1, 2003. New York: ACM, 2003. P. 267—273.

19. **Lee Daniel D. and Seung H.** Algorithms for Non-negative Matrix Factorization // In the Proceedings of NIPS. Denver, USA. 2000. MIT Press, 2000. P. 556—562.

20. **Nakagawa H. and Mori T.** Simple but Powerful Automatic Term Extraction Method // In the Proceedings of the Second International Workshop on Computational Terminology. 2002. Association for Computational Linguistics, 2002. P. 29—35.

21. **Ding C., Li T., Peng W.** On the equivalence between Non-negative Matrix Factorization and Probabilistic Latent Semantic Indexing // Computational Statistics and Data Analysis. 2008. № 52. P. 3913—3927.

22. **Phan X.-H., Nguyen C.-T.** GibbsLDA++: A C/C++ implementation of latent Dirichlet Allocation (LDA). 2007. URL: http://gibbslda.sourceforge.net/

23. **European** Parliament Proceedings Parallel Corpus 1996—2011. URL: http://www.statmt.org/europarl/

24. **EuroVoc.** Multilingual Thesaurus of the European Union. URL: http://eurovoc.europa.eu/drupal/

25. **Navigli R. and Velardi P.** Semantic Interpretation of Terminological Strings // In the Proceedings of the 6th International Conference on Terminology and Knowledge Engineering. Nancy. August 28—30, 2002. INRIA. 2002. P. 95—100.

26. **Weka 3.** Data Mining Software in Java. URL: http://www.cs.waikato.ac.nz/ml/weka

27. **Manning C. D., Raghavan P. and Schutze H.** Introduction to Information Retrieval. Cambridge University Press, 2008.

28. **British** National Corpus. URL: http://www.natcorp.ox.ac.uk/

29. **Peñas A., Verdejo V., Gonzalo J.** Corpus-based Terminology Extraction Applied to Information Access // In the Proceedings of the Corpus Linguistics, 2001. Lancaster, UK. March 29—April 2, 2001. Lancaster: Lancaster University Press, 2001. P. 458—465.

30. **Basili R., Moschitti A., Pazienza M., Zanzotto F.** A Contrastive Approach to Term Extraction. In the Proceedings of the 4th Terminology and Artificial Intelligence Conference, 2001. URL: http://lrec.elra.info/proceedings/lrec2010/pdf/553_Paper.pdf

31. **Wong W., Liu W., Bennamoun M.** Determining Termhood for Learning Domain Ontologies using Domain Prevalence and Tendency // In the Proceedings of the 6th Australasian Conference on Data Mining. Gold Coast, Australia. December, 2007. Australasian Computer Society, 2007. P. 47—54.

32. **Kurz D. and Xu F.** Text Mining for the Extraction of Domain Retrieval Terms and Term Collocations // In the Proceedings of the International Workshop on Computational Approaches to Collocations. Vienna, Austria. July 22—23. 2002. URL: http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.86.7929&rep=rep1&type=pdf

33. **Park Y., Bird R. J., Boguraev B.** Automatic glossary extraction beyond terminology identification // In the Proceedings of the 19th International Conference on Computational Linguistics. Mexico City, Mexico. February 17—23, 2002. Howard International House, 2002. P. 1—7.

34. **Gelbukh A., Sidorov G., Lavin-Villa E., Chanona-Hernandez L.** Automatic Term Extraction using Log-likelihood based Comparison with General Reference Corpora // In the Proceedings of the Natural Language Processing and Information Systems. Cardiff, UK. June 23—25, 2010, Springer, 2010. P. 248—255.

35. **Nakagawa H. and T. Mori.** Automatic Term Recognition based on Statistics of Compound Nouns and their Components // Terminology. 2003. Vol. 9, N 2. P. 201—219.

36. **Nokel M. A., Bolshakova E. I., Loukachevitch N. V.** Combining Multiple Features for Single-Word Term Extraction // Компьютерная лингвистика и интеллектуальные технологии. По материалам конференции Диалог-2012. Бекасово, Московская область. 30 мая—3 июня, 2012. М.: Издательство РГГУ, 2012. С. 490—501.

УДК 004.89

А. А. Алексеев, аспирант, Московский государственный университет имени М. В. Ломоносова,
e-mail: a.a.alekseevv@gmail.com

Тематический анализ новостного кластера как основа для автоматического аннотирования

Предлагается метод построения групп семантически близких слов и выражений, описывающих различных участников сюжета новостного кластера — тематических узлов. Метод основан на совместном использовании разнородных факторов схожести, таких как структурная организация новостных кластеров, анализ контекстов вхождения языковых выражений, а также информацию из предопределенных ресурсов. В качестве базиса для извлечения многословных выражений и построения тематических узлов используются контексты слов. Оценка предложенного метода проводится на основе применимости полученной структуры к задаче построения обзорных рефератов новостных кластеров.

Ключевые слова: *искусственный интеллект, компьютерная лингвистика, автоматическое аннотирование, извлечение информации*

A. A. Alekseev

Thematic Analysis of a News Cluster as a Basis for Automatic Summarization

In this paper we consider a method for extraction of various references of a concept or a named entity mentioned in a news cluster. The method is based on joint applying of heterogeneous similarity features, such as the structural organization of news clusters, comparison of various word contexts, and information from predefined resources. The word contexts are used as basis for multiword expression extraction and main entity detection. At the end of cluster processing groups of thematically-related elements are obtained, in which the main element of a group is determined. Evaluation of the proposed algorithm is performed in news cluster summarization task.

Keywords: *artificial intelligence, computational linguistics, automatic summarization, information extraction*

Введение

Современные технологии автоматической обработки новостных потоков основаны на тематической кластеризации новостных сообщений, т. е. выделении совокупностей новостей, посвященных одному и тому же событию — новостных кластеров [1].

Кластер документов должен соответствовать ситуации или совокупности связанных ситуаций (основная тема кластера) [1, 2]. В описываемой ситуации есть набор участников, которые в исходном кластере:

— могут быть выражены не только словами, но и словосочетаниями;

— могут выражаться не одним, а совокупностью различных выражений.

Так, *акции некоторой компании* могут выражаться в текстах одного новостного кластера как собственно *акции компании, контрольный пакет акций, контрольный пакет, акционер компании, владелец компании, состав владельцев* и другие варианты именования.

Можно предположить, что качественное выделение участников ситуации, включая различные варианты их наименования в различных документах кластера, может помочь лучше определять основную тему новостного кластера, и, тем самым, повысить качество различных операций с новостными кластерами, таких как автоматическое аннотирование, определение новизны информации и др.

В работе предлагается модель представления содержания новостного кластера, описывающая основ-

ных участников ситуации с учетом вариативности их именования — ***тематическое представление*** новостного кластера. ***Тематический анализ*** — это автоматический процесс построения тематического представления. Рассматриваются методы улучшения качества извлечения основных участников новостного события, что включает нахождение совокупности слов и выражений, с помощью которых тот или иной значимый участник события именовался в документах новостного кластера. Предлагаемый метод основан на совместном использовании совокупности факторов, в том числе разного рода контекстов употребления слов в документах кластера, информации из предопределенных источников (тезаурус русского языка), а также особенностях построения текстов на естественном языке.

Статья организована следующим образом: после обзора существующих подходов, приведенного в разд. 1, в разд. 2 обсуждается теоретическая основа предлагаемого алгоритма, в частности, модель связанного текста. Подробное описание предлагаемого алгоритма и интеграция результатов его работы в методы автоматического аннотирования представлены в разд. 3 и 4 соответственно. Оценка полученных результатов проведена в разд. 5. Все примеры взяты из новостного кластера, посвященного смене руководства алмазодобывающей компании "Алроса", содержащего 12 новостных документов.

1. Обзор существующих методов

Задача определения вариативности именования в текстах является актуальной для различных задач автоматической обработки естественного языка. С формальной точки зрения она является задачей группирования набора языковых выражений входной текстовой коллекции на тематические группы, относящиеся к одинаковым сущностям. Существует ряд различных подходов со схожими постановками задач, наиболее близкими из которых являются следующие:

- построение лексических цепочек;
- вероятностные тематические модели (в частности, LDA).

Лексическая цепочка представляет собой последовательность семантически связанных слов (повторы, синонимы, гипонимы, гиперонимы и др.), что является известным подходом к моделированию связности текста на естественном языке [3—5]. Алгоритмы построения лексических цепочек основаны на использовании информации о связях между словами и выражениями, описанными в некотором заранее определенном ресурсе, например, тезаурусе английского языка WordNet и тезаурусе русского языка РуТез.

Использование лексических цепочек является идейно наиболее близким из существующих подходов к предлагаемому в данной статье подходу. Основные отличия предлагаемого подхода заключаются в расширении рассмотрения с одного документа на коллекцию документов, а также в использовании совокупности различных факторов для группировки семантически близких слов и выражений (а не только информации, описанной в предопределенном ресурсе).

Вероятностные тематические модели, такие как Latent Dirichlet Allocation [2, 6, 7], основаны на предположении, что документы на естественном языке являются комбинацией различных тем (топиков), в то время как каждая тема (топик) является вероятностным распределением над словами. Подобный статистический вывод обычно не учитывает информацию о существующих лексических отношениях между словами и внутреннем устройстве текстов на естественном языке. Результаты работы алгоритмов, основанных на подобных моделях, имеют вероятностный результат и трудно интерпретируемы.

2. Тематическое представление

Как известно, текст обладает такими свойствами, как глобальная и локальная связности. Глобальная связность текста проявляется в том, что его содержание может быть представлено в виде иерархической структуры пропозиций [8]. Самая верхняя пропозиция представляет собой основную тему документа, а пропозиции нижних уровней представляют собой локальные или побочные темы документа.

Пропозиция основной темы документа, т. е. взаимоотношения участников основной темы, должна находить свое отражение в конкретных предложениях текста, которые должны раскрывать и уточнять взаимоотношения между тематическими элементами. Если текст посвящен обсуждению взаимоотношений между тематическими элементами C_1, ..., C_n, то в предложениях текста должны обсуждаться детали этих отношений. Это проявляется в том, что сами тематические элементы C_1, ..., C_n или их лексические представители должны встречаться как разные актанты одних и тех же предикатов в конкретных предложениях текста.

Исходя из данных идей, для выявления участников ситуации, описываемой в исходном новостном кластере, сделаем ряд следующих предположений:

1) взаимодействие участников описывается в предложениях текста, поэтому, чем чаще слова (или выражения) встречаются в одних и тех же предложениях текста, тем больше вероятность того, что эти слова (или выражения) относятся к разным участникам ситуации;

2) каждому участнику в тексте соответствует группа слов и выражений; предполагается, что в тексте имеются наиболее частотное (главное название участника) и разные варианты, поэтому группа слов и выражений, относящихся к одному участнику, строится в форме узла, т. е. главное выражение и относящиеся к нему выражения — ***тематический узел***;

3) тематическое представление в предлагаемом подходе представляет собой совокупность выявленных тематических узлов и отношений между ними [4, 9]; под отношениями между тематическими узлами по-

нимается степень близости по используемым характеристикам схожести.

Данные предположения основаны на внутреннем устройстве и тематической структуре текстов на естественном языке [3, 8]. Более подробная информация о сделанных предположениях, а также описание проведенных экспериментов по проверке сделанных гипотез, представлены в работе [10]. Новостной кластер не является связным текстом, но посвящен одной ситуации (или совокупности связанных ситуаций) и содержит большое число документов, что влечет за собой усиление всех статистических особенностей.

3. Алгоритм построения тематического представления

3.1. Контексты употребления слов

Важным фактором для построения тематического представления являются контексты, в которых употребляются слова и выражения. Для получения контекстов слов предложения разбивают на фрагменты между знаками препинания и выделяют следующие типы контекстов в рамках таких фрагментов: вхождения рядом (*Near*), через глагол (*AcrossVerb*) и появление в рамках одного фрагмента (*NotN*). Кроме того, для всех прилагательных и существительных запоминают слова, встречающиеся в соседних предложениях (*NS*). Предложения для вычисления этого показателя берут не полностью, учитывают фрагменты предложений с начала и до фрагмента, содержащего глагол (включительно), что позволяет извлекать из соседних предложений наиболее значимые слова.

Числовые значения для типов контекстов вычисляются как число появления рассматриваемой пары слов или выражений в соответствующих контекстах внутри всего входного новостного кластера.

3.2. Сборка многословных выражений

Важной основой извлечения многословного выражения из текста документа является частотность его встречаемости в тексте. Однако кластер представляет собой структуру, в которой многие цепочки слов повторяются многократно. Поэтому основным критерием для выделения многословных выражений является значительное превышение встречаемости слов непосредственно рядом друг с другом по сравнению с раздельной встречаемостью во фрагментах предложений [11]:

$$Near > 2(AcrossVerb + NotN).$$

Просмотр подходящих пар слов (выражений) для склейки проводится в порядке снижения коэффициента $Near/(AcrossVerb + NotN)$. При нахождении подходящей пары слов они склеиваются в единый объект, и все контекстные отношения пересчитывают. Процедура просмотра начинается заново и повторяется до тех пор, пока проведена хотя бы одна склейка.

В результате данной процедуры собираются такие выражения, как *президент компании, международные экономические отношения, председатель совета директоров, контрольный пакет акций* и т. д.

3.3. Характеристики для определения семантических связей

Для определения семантически связанных выражений и последующего построения тематических узлов используется набор из пяти основных характеристик схожести. Некоторые из данных характеристик являются контекстно-зависимыми и их вычисляют непосредственно на основании рассматриваемого новостного кластера, в то время как другие определяют на основании формальной схожести выражений и информации из заранее известных ресурсов. Каждая характеристика добавляет некоторый балл в общий вес схожести пары выражений, независимо от других характеристик схожести. Далее дано подробное описание алгоритма расчета весов схожести пар выражений.

Контекстно-зависимые характеристики. *Число вхождений в соседние предложения* (*Neighboring Sentence Feature*, *NSF*). Данная характеристика основана на гипотезе глобальной связности текстов на естественном языке [8] и ее следствии о том, что элементы одного тематического узла чаще появляются в соседних предложениях исходных документов, чем в одних и тех же предложениях.

Характеристика *NSF* вычисляется на основе контекстных параметров *AcrossVerb*, *Near*, *NotNear* и *NS* и распределения их средних значений внутри исходного новостного кластера. Характеристика *NSF* дает численную оценку соотношения количества вхождений в соседние предложения (характеристика *NS*) по отношению к числу вхождений в одни и те же предложения исходного корпуса (характеристики *AcrossVerb*, *Near* и *NotNear*) и основана на следующем соотношении:

$$W = NS - 2(AcrossVerb + Near + NotNear).$$

Общая формула вклада характеристики *NSF* в вес схожести пары выражений имеют следующую форму:

$$NSF = \mathrm{Min}\left[0{,}5,\ \frac{W}{\mathrm{Avg}(W)}\right],$$

где $\mathrm{Avg}(W)$ является средним значением W среди всех положительных значений в рамках всего кластера.

Характеристика *NSF* также является управляющей характеристикой. Это означает, что два выражения не могут быть включены в один и тот же тематический узел, если значение характеристики *NSF* отрицательное. Стоит отметить, что подобная характеристика не использовалась раньше для задач таких типов, как определение вариантов именования основных участников ситуации, построение рядов квазисинонимов, а также лексических цепочек.

Строгие контексты (*Strict Context*, *SC*). Данная характеристика основана на сравнении строгих контекстов употреблений слов — текстовых шаблонов. В ка-

Таблица 1

Пример ранжирования пар-кандидатов

Пары	Характеристики					Вес
	Контекстно-независимые		Контекстно-зависимые			
	BS	*TS*	*NSF*	*SC*	*SPS*	
Президент России — Президент РФ	0,90	1,00	0,00	0,50	0,34	**2,74**
Инвестгруппа — Инвестиционная группа	0,90	1,00	0,20	0,00	0,32	**2,42**
ГМК Норильский никель — Норильский никель	1,00	1,00	0,20	0,00	0,11	**2,31**
Российская Федерация — Россия	0,90	1,00	0,00	0,00	0,25	**2,15**
Отставка — Отставка с должности	0,90	1,00	0,20	0,00	0,00	**2,10**

честве шаблонов рассматривают 4-граммы: по два слова справа и слева от рассматриваемого выражения. Чем больше одинаковых шаблонов разделяет пара-кандидат, тем больше схожесть по данной характеристике. Вес шаблона строгого контекста рассчитывается следующим образом: каждое слово *n*-граммы шаблона контекста имеет вес, равный 0,25. Например, *n*-грамма (***, ***, *состоит, из*) будет иметь вес 0,5, а *n*-грамма (*новостной, кластер, состоит, из*) будет иметь вес, равный 1,0, что является максимальным весом полного шаблона *n*-граммы.

Значение характеристики *SC* вещественное и принадлежит отрезку [0, 1]. Вес характеристики вычисляется относительно веса пары с максимальным значением разделяемых строгих контекстов, пропорционально весу разделяемых строгих контекстов для текущей пары.

Схожесть контекстов употребления по внутренним характеристикам предложения (*Scalar Product Similarity*, *SPS*). Каждый из контекстных параметров, описанных в разд. 3.1, представляет собой вектор частот, сопоставленных с каждым словом или выражением. После построения данные контекстные векторы могут быть поставлены в соответствие классическим метрикам схожести, например, косинусной мере угла между векторами. Характеристика *SPS* может быть рассмотрена как более сглаженная и гибкая характеристика по отношению к характеристике *SC*, так как обе данные характеристики основаны на контекстах употребления слов и выражений.

Значение характеристики *SPS* вещественное и лежит в пределах от 0 до 0,5 (половинный вес характеристики), оно вычисляется как косинусная мера схожести по всем контекстным характеристикам (*AcrossVerb*, *Near* и *NotNear*), ограниченная сверху значением 0,5.

Контекстно-независимые характеристики. *Формальное сходство* (*Beginning Similarity*, *BS*). Рассмотрение формального сходства выражений является естественным путем обнаружения семантически связанных объектов. На текущий момент используется простая метрика схожести — одинаковые начала слов. Данная характеристика позволяет находить сходство между такими выражениями, как *руководитель — руководство, Президент России — Российский президент* и т. д.

Общий вес характеристики *BS* имеет вещественное значение из отрезка [0, 1] и вычисляется по следующей формуле (в случае, если есть слова с одинаковыми началами, иначе вес равен нулю):

$$BS = 1{,}0 - 0{,}1N_{diff},$$

где N_{diff} — число слов с различными началами.

Информация о схожести, описанная во внешнем ресурсе — тезаурусе РуТез (*Thesaurus Similarity*, *TS*). Вычисление характеристики *TS* основано на использовании информации из тезауруса русского языка РуТез [12]. Рассматривают различные типы связей (синонимия, часть — целое, род — вид), а также как непосредственные связи объектов, так и "длинные" связи по транзитивным типам отношений.

Значение характеристики *TS* вещественное в пределах от 0 до 1 и вычисляется обратно-пропорционально расстоянию между объектами:

$$TS = 1{,}0 - 0{,}2N_{rel},$$

где N_{rel} — длина пути по отношениям тезауруса (число связей).

Общий вес схожести пары рассматриваемых объектов вычисляется как сумма весов по отдельным характеристикам схожести, описанным выше. Таким образом, каждая пара получает вес, лежащий в пределах от 0 (отсутствие схожести) до 5 (максимальная схожесть), получаемый на основе шести характеристик (три контекстно-зависимых и три контекстно-независимых), лежащих в пределах от 0 до 1 (*SC*, *BS*, *TS*) и от 0 до 0,5 (*NSF*, *SPS*). Пример ранжирования пар в соответствии с описанным алгоритмом приведен в табл. 1 (топ-5 пар по общему весу).

3.4. Алгоритм построения тематического представления на основе совокупности факторов

Алгоритм построения тематического представления конструирует тематические узлы из пар выражений в порядке убывания их схожести. Предлагаемая структура тематического узла обладает следующими свойствами:

- текстовое выражение может принадлежать к одному или двум тематическим узлам; разрешение множественной принадлежности обеспечивает возмож-

ность представления различных аспектов исходного текстового выражения, а также его лексической многозначности;

- каждый тематический узел имеет главный элемент — центр тематического узла, который может принадлежать только к одному тематическому узлу; центр тематического узла является наиболее частотным элементом среди всех элементов тематического узла.

Построение тематического представления происходит на основе следующих принципов:

- рассматривается пара текстовых выражений с наибольшим весом схожести среди всех пар-кандидатов;
- более частотный элемент пары поглощает менее частотный элемент вместе со всеми его текстовыми вхождениями и контекстными характеристиками и становится представителем данной пары текстовых выражений — центром нового тематического узла;
- менее частотный элемент рассматриваемой пары может в дальнейшем аналогичным образом присоединиться к другому тематическому узлу;
- объединение тематических узлов, состоящих из нескольких элементов, происходит аналогично объединению одиночных текстовых выражений; центр более частотного тематического узла становится центром нового, объединенного тематического узла.

Итеративный процесс продолжается до тех пор, пока есть пары-кандидаты для объединения с весом схожести выше заданного порога. Например, тематический узел с центральным элементом *Пост* проходит следующие этапы в процессе построения (показаны пары с максимальным весом схожести на разных итерациях; более частотный элемент пары является первым элементом).

Итерация 7: (***Отставка***) ← (***Отставка с должности***)

Итерация 33: (***Отставка,*** *Отставка с должности*) ← ← (***Уход в отставку***)

Итерация 44: (***Отставка,*** *Отставка с должности, Уход в отставку*) ← (***Отставка президента***)

Итерация 61: (***Уход с поста***) ← (***Уход в отставку***)

Итерация 62: (***Отставка,*** *Отставка с должности, Уход в отставку, Отставка президента*) ← (***Уход с поста,*** *Уход в отставку*)

Итерация 102: (***Отставка,*** *Отставка с должности, Уход в отставку, Отставка президента, Уход с поста*) ← (***Пост***)

Итерация 103: (***Пост,*** *Отставка, Отставка с должности, Уход в отставку, Отставка президента, Уход с поста*) ← (***Должность***)

Итерация 104: (***Пост,*** *Отставка, Отставка с должности, Уход в отставку, Отставка президента, Уход с поста, Должность*) ← (***Уход***)

Следующие тематические узлы были получены в результате работы описанного алгоритма для кластера примера. Представлены пять наиболее частотных тематических узлов в порядке убывания частоты. Данные узлы не подвергались какой-либо постобработке, центры тематических узлов выделены полужирным шрифтом.

Пост: *уход с поста; должность; уход; отставка; отставка с должности; уход в отставку; отставка президента*

Алроса: *президент Алроса; АК Алроса*

Компания: *акция компании; владелец компании; объединение компаний; акция; акционер компании; владелец; пакет акций; состав владельцев; контрольный пакет акций; контрольный пакет; владение*

Ничипорук: *Александр Ничипорук*

Якутия: *президент Якутии; якутский; якутский президент*

4. Порождение аннотаций на основе тематического представления

Тематическое представление содержит в себе дополнительную информацию о внутреннем устройстве исходного новостного кластера, которая может быть использована для улучшения автоматических операций над текстовыми данными. Одной из таких задач является задача автоматического аннотирования, т. е. подготовки краткого изложения содержания исходного документа(ов). Решение данной задачи в значительной степени связано с наличием информации о различном именовании одних и тех же участников ситуации, описанной во входных документах, так как практически невозможно построить полную и неизбыточную аннотацию без учета вариативности упоминаний наиболее значимых сущностей. В данном разделе описаны как существующие известные методы аннотирования (*MMR*, *SumBasic*), так и новые подходы, основанные на тематическом представлении. Все представленные подходы используют для оценки качества построенного тематического представления путем его интеграции в исходную структуру данных алгоритмов аннотирования.

4.1. Метод Maximal Marginal Relevance

Метод *Maximal Marginal Relevance* (*MMR*) для задачи многодокументного аннотирования является классическим не порождающим (выбирающим для аннотации целые предложения из исходных документов) методом аннотирования, который основан на концепции *Maximal Marginal Relevance* для информационного поиска [13]. В оригинале данный алгоритм является запрос-ориентированным, но существует также вариант и для общего аннотирования, когда в качестве запроса для аннотирования выступает исходных корпус документов.

Критерий *MMR* заключается в том, что лучшее предложение для аннотации должно быть максимально релевантным исходному набору документов и максимально отличным от всех предложений, уже отобранных в итоговую аннотацию.

Аннотация строится итеративно на основании списка ранжированных предложений. Предложение с максимальным значением *MMR* выбирается на каждой итерации алгоритма:

$$MMR = \arg\max_{s \in S} [\lambda Sim_1(s,\ Q) - (1 - \lambda) \max_{s_j \in E} Sim_2(s,\ s_j)],$$

где S — множество предложений-кандидатов в аннотацию; E — множество отобранных в аннотацию; Q — запрос для аннотирования (входная коллекция в случае обзорного аннотирования); λ — интерполяционный коэффициент между релевантностью отбираемых пред-

ложений и их не избыточностью; Sim_1 — метрика схожести между предложением и запросом для аннотирования (например, косинусная мера угла между векторами, широко применяемая в информационном поиске); Sim_2 может быть такой же как Sim_1, или другой метрикой схожести. В данной работе в качестве метрик Sim_1 и Sim_2 использовалась косинусная мера угла между векторами.

4.2. Метод SumBasic

SumBasic — алгоритм для общего многодокументного аннотирования [14, 15]. В его основе лежит эмпирическое наблюдение о том, что более частотные слова рассматриваемого кластера документов с большей вероятностью попадают в экспертные аннотации, нежели слова с низкой частотностью.

Алгоритм *SumBasic* строится на базе частотного распределения слов в исходном документе и состоит из пяти шагов. На первом шаге происходит расчет вероятностей слов исходного кластера $p(w_i)$:

$$p(w_i) = n/N,$$

где n — число появлений слова w_i в исходной коллекции, N — общее число слов в данной коллекции. Каждому предложению S_j на втором шаге назначается вес, равный средней вероятности слов в данном предложении:

$$weight(S_j) = \Sigma w_i \in S_j \frac{p(w_i)}{\left|\{w_i \middle| (w_i \in S_j)\}\right|}.$$

На третьем шаге предложение с наибольшим весом отбирается в итоговую аннотацию. После этого на четвертом шаге происходит пересчет вероятностей всех слов, входящих в отобранное предложение, по следующей формуле:

$$p_{new}(w_i) = p_{old}(w_i)p_{old}(w_i),$$

где $p_{old}(w_i)$ и $p_{new}(w_i)$ — вероятности слова w_i до и после пересчета соответственно. На пятом шаге проверяется общая длина получившейся аннотации, и если она не превосходит заданного порога, то происходит переход ко второму шагу.

4.3. Собственные методы аннотирования на основе тематического представления

Документы новостного кластера содержат в себе описание некоторого события (ситуации) или ряда связанных событий (ситуаций). Основная цель автоматического аннотирования заключается в наиболее полном отражении значимых фактов, относящихся к данному событию (ситуации) и описанных в исходной коллекции документов. Событие (ситуация), в свою очередь, характеризуется набором ее участников. Под "фактом" в данном случае понимают описание взаимоотношений между некоторыми участниками события (ситуации) или же детализацию описания отдельного ее участника.

Тематический узел предлагаемого тематического представления является воплощением некоторого участника события (ситуации) и в идеале должен содержать всевозможные варианты именования данного участника в рамках исходного новостного кластера. Таким образом, в основе предлагаемых методов аннотирования лежит учет наиболее значимых взаимоотношений тематических узлов построенного тематического представления — учет взаимоотношений основных участников события (ситуации). Предлагается два итеративных метода аннотирования, отличающихся стратегией учета значимости отношений между тематическими узлами. На каждой итерации в обоих подходах отбирается по одному предложению.

В качестве основы для расчета значимости отношений между тематическими узлами в первом алгоритме (*OurSummary_Nodes*) выступает значимость самих тематических узлов. Каждый тематический узел имеет вес, равный суммарной частоте его элементов. На каждой итерации в итоговую аннотацию отбирается предложение, содержащее три наиболее значимых и еще не упомянутых тематических узла (ТУ_*NEW*):

$$s_i \Rightarrow \max\left(\sum_{\text{ТУ}\_NEW_j \in s_i,\ i=1\ldots3}^{desc\ weight(\text{ТУ}\_NEW_j)} weight(\text{ТУ}\_NEW_j)\right),$$

где *desc weight*(ТУ_*NEW*) обозначает рассмотрение тематических узлов в порядке убывания их веса — первых трех наиболее значимых узлов.

В рамках второго предлагаемого алгоритма аннотирования (*OurSummary_Relations*) критерием для отбора предложения выступает наличие наиболее обсуждаемой и еще не упомянутой пары тематических узлов. Для каждой пары тематических узлов предварительно рассчитывается число вхождений в одни и те же предложения исходного новостного кластера — "обсуждаемость" пары. На каждой итерации отбирается предложение, содержащее наиболее обсуждаемую и неупомянутую в отобранных предложениях пару, а также обладающее наибольшим общим весом тематических узлов (ТУ_*REL_NEW* — неупомянутая пара тематических узлов):

$$s_i \Rightarrow$$
$$\Rightarrow \max\left(\sum_{\text{ТУ}\_REL\_NEW_j \in s_i} weight(\text{ТУ}\_REL\_NEW_j)\right).$$

Итеративный процесс отбора предложений для аннотации в обоих алгоритмах продолжается до тех пор, пока не будет превышен заданный порог по числу слов. Во всех генерируемых аннотациях данный порог равен 100 словам — стандартный размер аннотации для подобных задач на соревнованиях мирового уровня (*Document Understanding Conference, Text Analysis Conference*).

5. Оценка качества аннотаций

5.1. Методы оценки автоматических аннотаций ROUGE и Пирамид

Оценка качества порождаемых аннотаций является достаточно сложной процедурой. Несомненно, наиболее правдоподобные оценки можно получить при

помощи ручной оценки путем привлечения большого числа экспертов. Но данный метод является очень дорогим и трудоемким. Поэтому используют автоматические методы оценки качества аннотаций ROUGE [16] и формализованный метод Пирамид.

Метод ROUGE основан на автоматическом сравнении порожденной аннотации с эталонными аннотациями, созданными экспертами. Существуют разлные модификации алгоритма, связанные с разными способами сравнения: сравнение *n*-грамм (ROUGE-N; ROUGE-1 — монограммы, ROUGE-2 — биграммы); сравнение максимальных общих последовательностей (ROUGE-L и ROUGE-W); сравнение пропусков монограмм и биграмм (ROUGE-S и ROUGE-SU). В работе [16] показано, что все основные ROUGE-метрики являются значимыми, так как в зависимости от специфики конкретной задачи каждая из метрик может иметь наилучшую корреляцию с ручными аннотациями.

В основе метода Пирамид также лежит сравнение автоматических аннотаций с эталонными аннотациями. Но в отличие от метода ROUGE данное сравнение происходит не в автоматическом режиме, а в ручном, на основании формализованного алгоритма сравнения. Эксперты выделяют из эталонных аннотаций все "информационные единицы" (*Summary Content Units*, *SCU*) — факты, описанные в аннотации. Каждая информационная единица получает вес пропорционально числу упоминаний в экспертных аннотациях. Далее полученные информационные единицы вручную ищутся в автоматических аннотациях. Итоговая оценка аннотации равна общему весу упомянутых информационных единиц по отношению к суммарному весу информационных единиц, извлеченных для данного новостного кластера.

В данной работе оценка качества аннотаций построена на оценке методом ROUGE и дополнительной оценке методом Пирамид.

5.2. Автоматические аннотации и их оценка

Для оценки качества автоматических аннотаций были подготовлены 11 новостных кластеров по различным тематикам (политика, спорт, происшествия и др.), собранные на основе пословной модели представления данных [1]. Независимые эксперты-лингвисты подготовили от двух до четырех ручных аннотаций для каждого из данных кластеров. Все автоматические аннотации прошли единообразную обработку для автоматической оценки программным пакетом ROUGE [16]: ограничение длины аннотации, лемматизация, исключение стоп-слов и т. д. Всего в оценке участвовали десять различных модификаций алгоритмов: четыре модификации *MMR* (с учетом (*_WithIDF*) и без учета (*_WithoutIDF*) *IDF*; с итерацией (+ *Groups*) и без интеграции тематического представления), две модификации SumBasic и четыре модификации новых методов аннотирования (*OurSummary_Nodes* и *OurSummary_Relations*).

5.3. Результаты

По итогам работы программного пакета ROUGE каждая автоматическая аннотация получает набор результатов по различным метрикам сопоставления автоматических аннотаций с аннотациями, составленными экспертами. По причине значимости различных ROUGE-метрик для задач с различной спецификой (см. подразд. 5.1) в качестве основного параметра для сравнения автоматических аннотаций была взята средняя позиция в результатах по всем основным ROUGE-метрикам.

В табл. 2 приведены итоговые результаты оценки всех исследуемых модификаций алгоритмов по всем основным ROUGE-метрикам, а также агрегирующая оценка, по которой выполнена сортировка. В скобках после оценки указывается позиция данного результата относительно других методов по данной метрике ROUGE.

Для проведения дополнительной оценки качества автоматических аннотаций лучших и наиболее значимых методов (с и без интеграции построенного тематического представления) была проведена альтернативная оценка полученных аннотаций методом Пирамид [17]. Результаты данной оценки представлены в табл. 3.

На основе результатов оценки полученных аннотаций методами ROUGE и Пирамид необходимо отметить, что:

Таблица 2

Результаты оценки автоматических аннотаций методом ROUGE

Метод	Мера ROUGE					Балл
	1	2	L	S	SU	
MMR + Groups	**0,62499 (1)**	**0,41633 (1)**	**0,6021 (1)**	**0,35529 (1)**	**0,36649 (1)**	**1,0**
OurSummary_Nodes	0,58652 (2)	0,36154 (3)	0,5645 (2)	0,32113 (2)	0,33203 (2)	2,2
OurSummary_Nodes_WithIDF	0,58497 (3)	0,33918 (5)	0,55745 (3)	0,30124 (3)	0,31283 (3)	3,4
MMR_WithIDF	0,57623 (4)	0,38116 (2)	0,55503 (4)	0,29792 (4)	0,30971 (4)	3,6
MMR_WithoutIDF	0,56784 (5)	0,34595 (4)	0,55124 (5)	0,26092 (5)	0,27349 (5)	4,8
OurSummary_Relations	0,53141 (6)	0,2892 (6)	0,50422 (6)	0,25382 (6)	0,26509 (6)	6,0
SumBasic + Groups	0,52255 (7)	0,22881 (9)	0,493 (8)	0,24356 (7)	0,25525 (7)	7,6
SumBasic	0,51847 (8)	0,24735 (8)	0,49786 (7)	0,23064 (8)	0,24257 (8)	7,8
OurSummary_Relations_WithIDF	0,45494 (9)	0,24856 (7)	0,43768 (9)	0,19419 (10)	0,20492 (10)	9,0
MMR_WithIDF + Groups	0,44475 (10)	0,22238 (10)	0,42318 (10)	0,20627 (9)	0,21648 (9)	9,6

Таблица 3

Результаты оценки методом Пирамид

Метод	Оценка по методу Пирамид
MMR + Groups	**0,645 (1)**
MMR_WithIDF	0,617 (2)
OurSummary_Nodes	0,602 (3)
SumBasic + Groups	0,575 (4)
SumBasic	0,567 (5)

- наилучший результат показал метод аннотирования, основанный на построенном тематическом представлении;
- добавление тематических узлов к обоим базовым методам улучшило результаты исходных методов.

Заключение

В статье предложен алгоритм выявления семантически связанных слов и выражений, описывающих различных участников ситуации новостного кластера — тематические узлы. Предложенный алгоритм основан на совместном использовании характеристик схожести различной природы. В дополнение к известным контекстным характеристикам схожести, таким как анализ жестких контекстов (шаблонов) употребления слов и выражений, используется характеристика, основанная на внутреннем устройстве текстов на естественном языке — анализ встречаемости в соседних предложениях кластера по отношению к встречаемости в одних и тех же предложениях. В едином алгоритме объединены характеристики следующих различных типов:

- формальное сходство слов и выражений;
- информация из предопределенных ресурсов (тезаурус РуТез [12]);
- контекстные характеристики схожести.

Оценка предложенного алгоритма проводилась в контексте применения полученного тематического представления к задаче автоматического аннотирования. Полученные результаты подтверждают, что информация, заложенная в построенных тематических узлах, позволяет улучшать качество алгоритмов многодокументного аннотирования.

Список литературы

1. **Добров Б. В., Павлов А. М.** Исследование качества базовых методов кластеризации новостного потока в суточном временном окне // Труды конференции RCDL'2010. 13—17 октября 2010, г. Казань. 2010. С. 287—295. URL: http://rcdl.ru/doc/2010/287-295.pdf

2. **Allan J.** Introduction to Topic Detection and Tracking // Topic detection and tracking, Kluwer Academic Publishers Norwell, MA, USA, 2002. P. 1—16.

3. **Hirst G., St-Onge D.** Lexical Chains as representation of context for the detection and correction malapropisms // WordNet: An electronic lexical database and some of its applications / eds. C. Fellbaum. Cambrige, MA: The MIT Press, 1998. P. 305—332.

4. **Loukachevitch N.** Multigraph representation for lexical chaining // Proceedings of SENSE Workshop 2009. Moscow, Russia. July 2009. P. 67—76.

5. **Лукашевич Н. В., Добров Б. В.** Исследование тематической структуры текста на основе большого лингвистического ресурса // Компьютерная лингвистика и интеллектуальные технологии: труды Международной конференции Диалог'2000, Протвино, Россия. 2000. С. 252—258.

6. **Blei D., Ng A., Jordan M.** Latent Dirichlet Allocation // Journal of Machine Learning Research. 2003. № 3. P. 993—1022.

7. **Griffiths T., Steyvers M.** Finding scientific topics // Proceedings of the National Academy of Sciences of the United States of America. 2004. Vol. 101, Suppl. 1. P. 5228—5235.

8. **Dijk van T. A.** Semantic Discourse Analysis // Handbook of Discourse Analysis / eds. Teun A. van Dijk, vol. 2. London: Academic Press, 1985. P. 103—136.

9. **Hasan R.** Coherence and Cohesive harmony // Understanding reading comprehension / eds. J. Flood. Newark, 1984. P. 181—219.

10. **Alekseev A., Loukachevitch N.** Use of Multiple Features for Extracting Topics from News Clusters // Труды конференции SYRCODIS'2012, Москва. Май, 2012. С. 3—11. URL: http://ceur-ws.org/Vol-899/paper2.pdf

11. **Добров Б. В., Лукашевич Н. В., Сыромятников С. В.** Формирование базы терминологических словосочетаний по текстам предметной области // Труды пятой всероссийской научной конференции "Электронные библиотеки: Перспективные методы и технологии, электронные коллекции". Санкт-Петербург. Октябрь 2003. С. 201—210. URL: http://rcdl.ru/doc/2003/F2.pdf

12. **Loukachevitch N., Dobrov B.** Evaluation of Thesaurus on Sociopolitical Life as Information Retrieval Tool // Proceedings of Third International Conference on Language Resources and Evaluation, Vol. 1. Spain, June 2002. P. 115—121. URL: http://www.lrec-conf.org/proceedings/lrec2002/pdf/188.pdf

13. **Carbonell J., Goldstein J.** The use of MMR, diversity-based reranking for reordering documents and producing summaries // Proceedings of the 21st Annual International ACM SIGIR Conference on Research and Development in Information Retrieval. Melbourne, Australia, 1998. P. 335—336. URL: http://www.cs.cmu.edu/~jgc/publication/The_Use_MMR_Diversity_Based_LTMIR_1998.pdf

14. **Nenkova A., Vanderwende L.** The impact of frequency on summarization. Microsoft Research Technical Report, MSR-TR-2005-101, 2005.

15. **Vanderwende L., Suzuki H., Brockett C., Nenkova A.** Beyond SumBasic: Task-focused summarization with sentence simplification and lexical expansion // Information Processing and Management Journal. 2007. Vol. 43, Is. 6. P. 1606—1618.

16. **Lin C.-Y.** ROUGE: a package for automatic evaluation of summaries // Proceedings of the Workshop on Text Summarization Branches Out (ACL'2004). Barcelona, Spain, 2004. P. 74—81. URL: http://acl.ldc.upenn.edu/acl2004/textsummarization/pdf/Lin.pdf

17. **Harnly A., Nenkova A., Passonneau R., Rambow O.** Automation of summary evaluation by the pyramid method // Proceedings of the International Conference on Recent Advances in Natural Language Processing. Borovets, Bulgaria. September 18—24, 2005. URL: http://www.cs.columbia.edu/~ani/papers/aabo-ranlp.pdf

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 09.01.2014 г. Подписано в печать 19.02.2014 г. Формат 60×88 1/8. Заказ PI314
Цена свободная.

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