электронный журнал
МОЛОДЕЖНЫЙ НАУЧНО-ТЕХНИЧЕСКИЙ ВЕСТНИК
Издатель ФГБОУ ВПО "МГТУ им. Н.Э. Баумана". Эл No. ФС77-51038.

УДК 004.45

Утилита удаленного управления программным обеспечением для UNIX-систем

***Мазалов А.А.**, студент*
Россия, 119296, г. Москва, МГТУ им. Н.Э. Баумана,
кафедра «Системы обработки информации и управления»

Научный руководитель: Виноградов В.И., к.т.н, доцент
Россия, , г. Москва, МГТУ им. Н.Э. Баумана
vinogradovs.fam@bmstu.ru

Введение

Программное обеспечение, используемое любой организацией, нуждается в эффективном управлении (администрировании). Такое управление означает учет имеющегося программного обеспечения (ПО), своевременное проведение работ по его установке, обновлению или удалению.

Вопрос обеспечения централизованного удаленного администрирования всех компьютеров организации приобретает особую актуальность в связи с массовой компьютеризацией рабочих мест. Наличие вычислительной сети позволяет администратору осуществлять централизованное управление программным обеспечением организации путем доступа со своего компьютера к удаленным автоматизированным рабочим местам (АРМ) сотрудников. Таким образом достигается значительное ускорение и оптимизация его работы.

Постановка задачи

Для централизованного администрирования программного обеспечения, установленного на удаленных компьютерах, требуется разработать программную систему (утилиту), выполняющую удаленное управление программным обеспечением на всех компьютерах, входящих в состав локальной сети организации. Программная система должна работать под управлением UNIX-подобных ОС, поддерживающих функционал RPM-пакетов. Требуется обеспечить централизованное управление компонентами ПО организации путем доступа с компьютера администратора к удаленным автоматизированным рабочим местам (АРМ) сотрудников. Необходимо следить за достоверностью передаваемой информации, а также ее защищенностью от злоумышленников. В качестве репозитория (хранилища) RPM-пакетов должен выступает

компьютер администратора. Доступ к Интернет-ресурсам для удовлетворения зависимостей (поиска недостающих компонентов ПО) не требуется.

Для реализации необходимого функционала следует решить две основные задачи: обеспечить управление программным обеспечением на локальном компьютере путем подачи ОС соответствующих команд, а также трансляцию этих команд на удаленные ЭВМ с использованием одного из сетевых протоколов.

Методы и средства управления ПО в UNIX-подобных ОС

Для автоматизации процесса управления ПО, установленным на компьютере, работающем под управлением UNIX-подобной операционной системы (ОС), можно либо воспользоваться базовыми средствами, предоставленными ОС, либо обратиться к одной из существующих программных разработок-дополнений, выполненных энтузиастами.

Базовые средства большинства UNIX-подобных ОС представлены системными утилитами управления программными пакетами. Формат программных пакетов может быть специфичным для конкретной ОС. Примерами программных пакетов могут служить RPM-пакеты, поддерживаемые такими дистрибутивами, как Fedora, Red Hat Enterprise Linux, Mandriva, Pacman-пакеты, предназначенные для дистрибутива Arch Linux и его вариантов, а также DEB-пакеты, предназначенные для дистрибутивов Debian и Ubuntu. Однако сущность этих программных пакетов одна и та же. Они представляют собой архивы, сформированные с помощью некоторого архивного контейнера (архиватора) и сжатые затем с помощью некоторой утилиты сжатия. В данных архивах собраны различные файлы программы, установочные скрипты, а также файл спецификации (называемый spec-файлом для RPM и control-файлом для DEB-пакетов), в котором содержатся инструкции для сборки пакета. Использование «пакетного подхода» к управлению программным обеспечением считается хорошей альтернативой сборке программ из исходных кодов с их последующим «ручным» разворачиванием на UNIX-станциях. Преимущества здесь заключаются в легкости управления программами (каждая операция производится с помощью одной единственной команды), возможности автоматической проверки целостности пакетов с помощью цифровых подписей, группирования нескольких программ, логически связанных друг с другом, в один пакет, обнаружения неудовлетворенных зависимостей при установке и обновлении, а также предотвращения краха системы, который может наступить при непроизвольном удалении некоторого программного компонента, без которого не могут работать другие программы. Системные утилиты, называемые также менеджерами пакетов, предоставляют

возможности установки, удаления и обновления программных пакетов. Их консольный интерфейс позволяет администратору компьютера гибко управлять ходом выполняемых действий с помощью многочисленных параметров и флагов.

Говоря о программных разработках-дополнениях, таких как YUM и URPMI для управления RPM-пакетами или APT для DEB-пакетов, следует отметить, что они призваны обеспечить более удобный интерфейс взаимодействия оператора с ОС, а также до некоторой степени расширить функционал базовых средств системы. На более низком логическом уровне работа таких программных средств предполагает обращение к базовым системным средствам.

Обзор возможностей RPM-утилиты

Задачу обеспечения управления программным обеспечением на локальном компьютере будем решать, используя системную утилиту управления RPM-пакетами. Выбор формата пакетов продиктован используемым дистрибутивом UNIX-подобной ОС (дистрибутив на основе Fedora). Термин RPM (англ. RPM Package Manager или Red Hat Package Manager) обозначает две сущности: формат пакетов программного обеспечения, а также программу, созданную для управления этими пакетами. Для хранения файлов в формате rpm используется архивный контейнер (архиватор) cpio, использующий сжатие с помощью утилиты gzip или иных утилит архивации. Утилита rpm, изначально входившая в состав сборок Red Hat, позднее была включена в состав большинства дистрибутивов UNIX. Системы семейства UNIX обеспечивают ведение специальной базы данных RPM, находящейся в каталоге /var/lib/rpm. Она состоит из одиночной базы данных (Packages), в которой хранится вся информация о пакетах, и множества малых баз (*__db.001*, *__db.002* и т. д.), которые служат для индексации и содержат в себе сведения о том, какие файлы менялись и создавались при установке и удалении пакетов. Проверка целостности пакетов производится с помощью контрольных сумм (например MD5) и GPG-подписей. Как уже отмечалось ранее, произвести установку программ из RPM-пакета, пользуясь одноименной системной утилитой, очень просто. Для этого достаточно подать в терминале команду:

```
rpm -i [flags] полное_имя_пакета.rpm
```

Аналогично, для обновления имеем:

```
rpm -U [flags] полное_имя_пакета.rpm
```

Для удаления:

```
rpm -e [flags] имя_пакета
```

Здесь [flags] — совокупность необязательных параметров-флагов, предназначенных для задания специфических режимов установки, удаления или обновления. Утилита rpm позволяет также с легкостью получить список уже установленных пакетов путем запроса к системной базе данных Packages:

rpm -qa

или

rpm -qi имя_пакета

Остановимся на правилах именования RPM-пакетов. Полное имя пакета, являющееся параметром запуска утилиты rpm с ключами "i", "U" и некоторых других, состоит из краткого имени, а также указания на версию, сборку пакета, а также архитектуру, под которую собран этот пакет. Формат полного имени RPM-пакета имеет вид:

<название>-<версия>-<релиз>.<архитектура>

Существование единых правил именования для RPM-пакетов, а также присутствие основной информации о пакете в его полном имени упрощает анализ полных имен пакетов, а также позволяет получить основную информацию о компонентах программного обеспечения без необходимости осуществления дополнительных запросов к операционной системе. Извлеченные из полного имени сведения могут быть в дальнейшем сохранены в специально созданные для хранения информации о пакете программные структуры данных.

Краткий обзор частичных аналогов разработанной системы

При решении задачи автоматизации управления компонентами ПО были использованы исключительно базовые средства операционной системы (менеджер пакетов rpm). Дело в том, что многочисленные дополнения к базовым средствам, такие как YUM (Yellow dog Updater, Modified), URPMI (User RPM Installer), о которых было упомянуто ранее, в основном частичными аналогами разрабатываемой утилиты в части автоматизации управления программными пакетами. Они отличаются способностью осуществлять поиск требуемых файлов и библиотек для удовлетворения неразрешенных зависимостей не только на локальном компьютере, но и в Интернете. С одной стороны, использование подобных дополнений могло бы обогатить функционал системы, а с другой — привести к его чрезмерному усложнению и повышенному расходу вычислительных ресурсов. Следует учесть также, что при использовании в условиях отсутствия доступа к Интернету или его ограниченности такой функционал оказывается

просто лишним. Отметим и то, что в отличие от разрабатываемой системы, они не имеют дружественного графического интерфейса пользователя (графический интерфейс предоставляется рядом отдельных разработок, например PackageKit и Yum Extender для YUM), а также тот факт, что даже с помощью вышеупомянутых средств не представляется возможным обеспечить централизованное управление удаленными компьютерами с компьютера администратора. Например, в случае необходимости осуществления таких действий, используя YUM, предлагается создать специальный конфигурационный файл с указанием имени репозитория и url-адреса компьютера, на котором находится этот репозиторий. Этот конфигурационный файл следует скопировать на все машины сети. Далее на каждом из администрируемых компьютеров следует выполнить команду "yum" с ключом "install". Таким образом, задача обеспечения трансляции команд по сети остается актуальной даже в случае использования программ типа YUM и URPMI.

Обзор возможностей протокола SSH

Для передачи команд и файлов по сети имеет смысл использовать протокол SSH (Secure Shell), поддержка которого реализована во всех UNIX-подобных системах. SSH представляет собой сетевой протокол прикладного уровня, обеспечивающий туннелирование TCP-соединений, то есть создание защищенного канала связи посредством инкапсуляции кадров TCP. Отличительной особенностью туннелирования является то, что инкапсулируемый протокол либо относится к тому же уровню модели OSI, что и инкапсулирующий, либо располагается ниже него в иерархии. С помощью SSH обеспечивается шифрование передаваемых данных и аутентификация пользователей по паролю или по ключу. SSH схож по функциональности с протоколами Telnet и Rlogin, однако шифрует весь передаваемый трафик, в том числе пароли. В настоящее время использование Telnet и Rlogin считается устаревшей и небезопасной практикой. Дело в том, что пароли в них передаются открытым текстом. Перехват этой информации злоумышленником означает получение контроля над всем процессом сетевой передачи.

Для работы по SSH требуется наличие SSH-сервера и SSH-клиента. Сервер прослушивает соединения от клиентских машин и при установлении связи производит аутентификацию, после чего начинает обслуживание клиента. Клиент используется для входа на удалённую машину и выполнения команд.

Аутентификация пользователя по ключу применяется как альтернатива аутентификации по паролю (хотя пароль может служить в качестве дополнительной меры

проверки подлинности при аутентификации по ключу). Многократный ввод пароля при осуществлении серии обращений к удаленной машине может быть утомителен или, в случае вызова утилиты "ssh" из прикладной программы, усложнить логику ее работы. Для организации ssh-аутентификации по ключу требуется сгенерировать ключевую пару, состоящую из открытого (public) и закрытого (private) ключей. По умолчанию для хранения ключевой пары создаются файлы "~/.ssh/id_rsa.pub" (для открытого ключа) и "~/.ssh/id_rsa" (для закрытого ключа). Генерация ключевой пары осуществляется командой:

ssh-keygen -t rsa

Здесь вместо "rsa" может применяться другой метод шифрования (например "dsa") в зависимости от версии протокола ssh.

Далее следует выполнить копирование файла открытого ключа на удаленный компьютер, к которому будет осуществляться доступ. Открытый ключ добавляется в файл "~/.ssh/authorized_keys". Редактирование этого файла можно осуществить вручную, однако рекомендуеся использовать команду:

ssh-copy-id user@server

Здесь:

"user" – польхователь, от имени которого производится подключение;

"server" – ip-адрес удаленного компьютера.

При первом подключении после создания ключевой пары будет задан вопрос о доверии ключу. В случае утвердительного ответа информация о скопированном открытом ключе добавляется в файл "~/.ssh/known_hosts".

Закрытый ключ не следует копировать на удаленные машины. Очевидно, что его нужно сохранять в тайне. При запросе аутентификации по ключу на удаленный компьютер посылается открытый ключ и запрашивается аутентификация по этому ключу. Если открытый ключ, полученный удаленным компьютером, присутствует в файле "~/.ssh/authorized_keys", с удаленного компьютера передается сообщение, зашифрованное этим ключом. Компьютер, инициирующий передачу, расшифровывает сообщение с помощью своего секретного ключа. Если расшифровка прошла успешно, пользователю предоставляется доступ к удаленному компьютеру.

Синтаксис удаленного вызова любой программы по протоколу SSH через терминал имеет вид:

ssh user@server команда

Здесь:

"user" – пользователь, от имени которого производится подключение;

"server" – ip-адрес удаленного компьютера.

"команда" – команда, транслируемая на удаленную машину по протоколу ssh.

Например, для удаления пакета "my_pack" с компьютера с ip-адресом 192.168.99.171 от имени пользователя "alex" следует выполнить команду:

ssh alex@192.168.99.171 rpm -e my_pack

В ходе реализации функционала по удаленной установке и обновлению RPM-пакетов была использована возможность копирования файлов на удаленный компьютер, предоставляемая SSH. Приведем синтаксис соответствующей команды:

scp путь_к_файлу user@server: удаленный_путь

Здесь:

"user" – пользователь, от имени которого производится подключение;

"server" – ip-адрес удаленного компьютера;

"путь_к_файлу" — полный путь к копируемому файлу на локальном компьютере;

"удаленный_путь" — полный путь к директории на удаленном компьютере, в которую должен быть скопирован файл.

Описание разработанной системы

Для программной реализации функционала удаленного управления компонентами ПО была разработана программа, выполняющая следующий перечень действий:

– Построение списка пакетов, находящихся на компьютере администратора в указанной директории;

– Построение списка компьютеров, входящих в состав локальной сети;

– Установка выбранных из списка пакетов на компьютеры, выбранные из списка доступных компьютеров сети;

– Обновление выбранных из списка пакетов на компьютерах, выбранных из списка доступных компьютеров сети;

– Удаление выбранных из списка пакетов с компьютеров, выбранных из списка доступных компьютеров сети;

– Поддержка основных опций установки, удаления и обновления пакетов.

– Журналирование операций, уведомление администратора об ошибках и результатах действий по удаленному администрирования компонентов ПО

При проектировании программы была разработана система классов, показанная на Рис. 1.

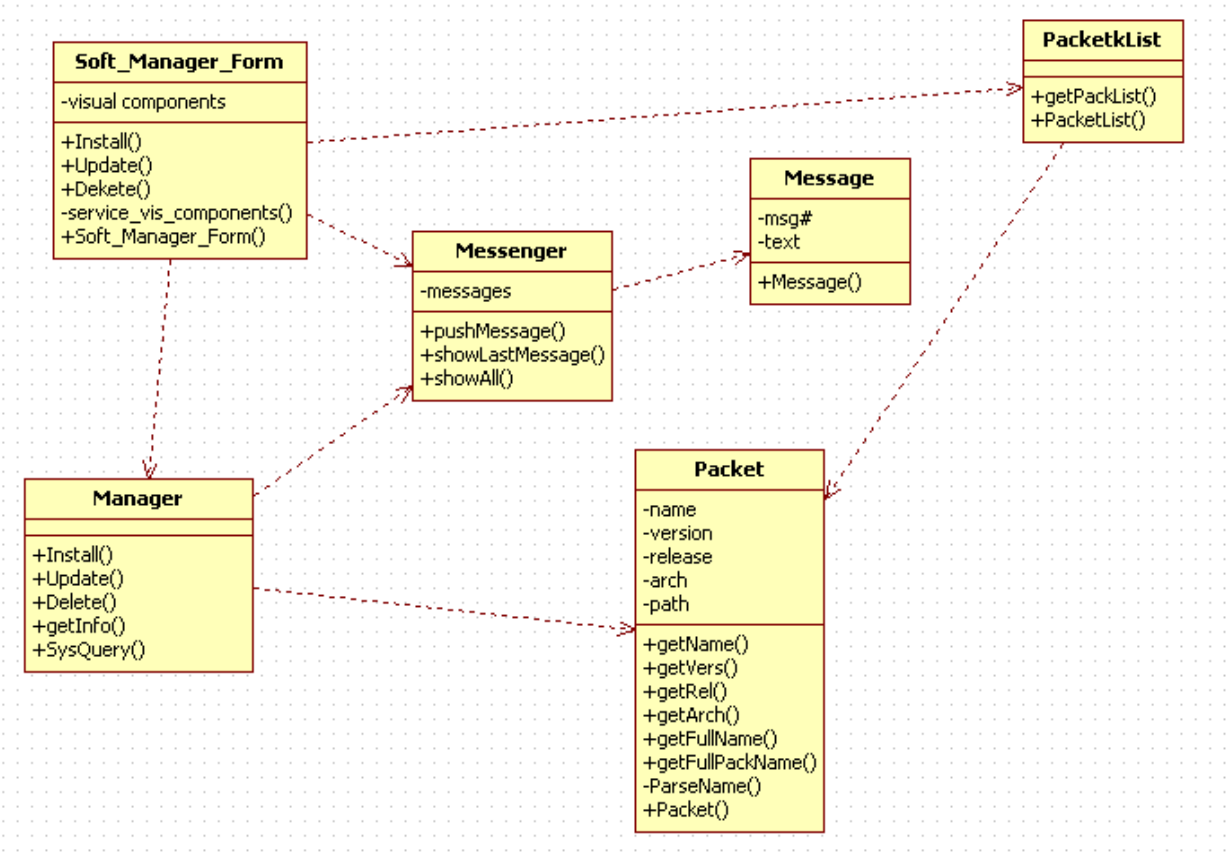

Рис. 1. Диаграмма классов разработанной системы

В состав системы классов входит класс RPM-пакета Packet, функциональный класс для управления RPM-пакетами Manager, класс списка RPM-пакетов PacketList, класс, отвечающий за генерацию и хранение сообщений об ошибках Messenger, а также класс Soft_Manager_Form, соответствующий графическому интерфейсу пользователя.

Класс RPM-пакета служит для хранения основной информации о компоненте программного обеспечения: имени пакета, пути к пакету, его версии, сборки, а также архитектуры, под которую собран данный пакет. Класс RPM-пакета содержит также методы для получения значений скрытых полей класса, конструктор, принимающий в качестве входного параметра имя RPM-пакета и выполняющий его разбор с помощью скрытого метода ParseName(). Данные, извлеченные из имени пакета, сохраняются затем как значения полей класса.

Класс списка RPM-пакетов содержит примитивный по функционалу конструктор, а также метод getPackList, выполняющий запрос информации о пакетах, расположенных в заданной директории на локальном компьютере и возвращающий полученные данные в виде списка (стандартного контейнера) экземпляров класса RPM-пакета.

Класс Manager предоставляет функционал для удаленной установки, удаления, обновления RPM-пакетов, а также получения информации о пакете с помощью соответствующих методов. Доступ к удаленным узлам сети выполняется по протоколу SSH. Класс Manager также позволяет осуществить запрос непосредственно к операционной системе с помощью метода SysQuery().

Класс Messenger обеспечивает журналирование ошибок, их первичную обработку и вывод пользователю. Он содержит список (стек) сообщений, а также методы помещения сообщений в стек, выборки последнего сообщения из стека, а также полного опорожнения стека сообщений. Основные ошибки, возникающие при работе с утилитой удаленного управления программным обеспечением, связаны с нарушением процесса сетевой передачи, попыткой удалить неустановленный пакет, а также наличием неудовлетворенных зависимостей при установке и обновлении или угрозы повреждения системы при попытке удалить пакет, входящий в список зависимостей других пакетов. При возникновении ошибки методы класса Manager помещают в стек новое сообщение. Элементами стека сообщений являются экземпляры класса Message (сообщение). Опорожнение стека выполняется в классе пользовательского интерфейса Soft_Manager_Form.

Класс пользовательского интерфейса содержит визуальные компоненты, с помощью которых пользователь может выбрать пакеты для установки/удаления/обновления, а также компьютеры сети, на которых следует выполнить удаленное управление пакетами. Главная форма системы представлен на Рис. 2.

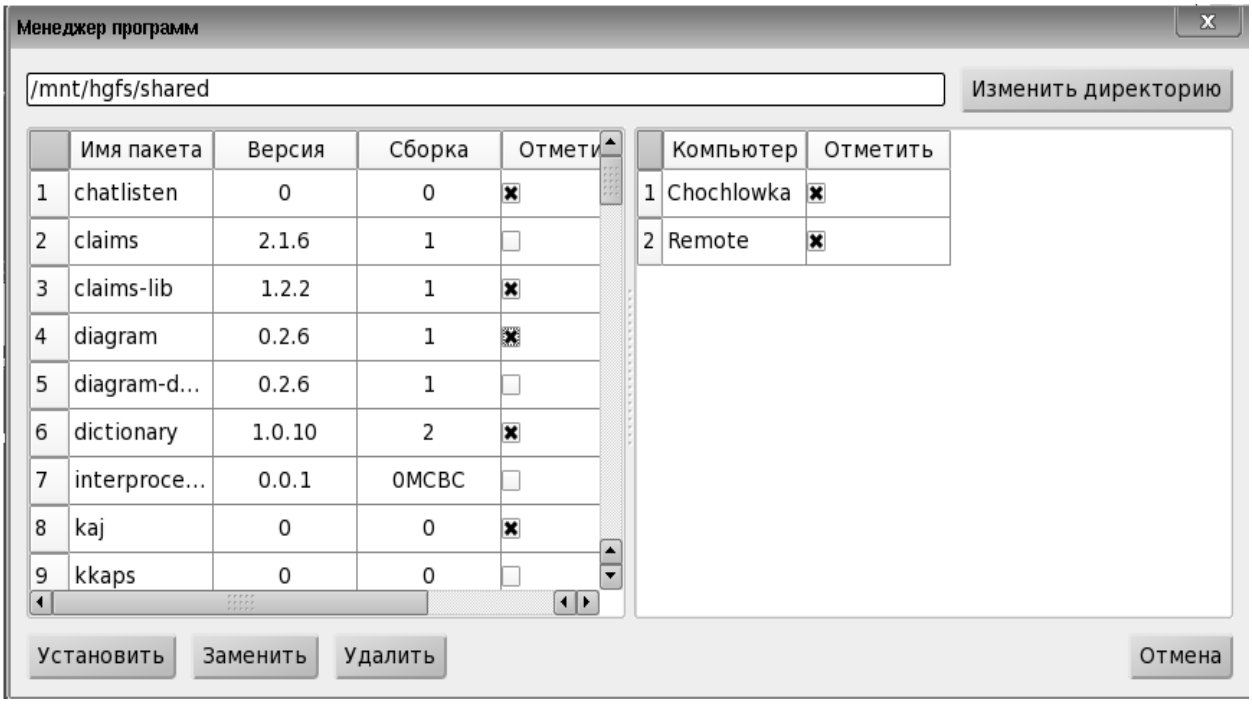

Рис. 2. Главная форма системы

Программная система разрабатывалась на языке C++ в среде Qt. Для управления данными использовалась СУБД PostgreSQL. Преимуществами обоих продуктов является их распространение на правах свободного ПО, а также поддержка широкого спектра возможностей, повышающих удобство взаимодействия с БД и программирования.

Заключение

В результате проведенной работы были проанализированы возможности, предоставляемые UNIX-подобными ОС для управления ПО. Были рассмотрены механизм администрирования ПО с помощью RPM-пакетов, сетевой протокол SSH и показано, как с использованием данного протокола можно выполнять централизованное управление удаленным компьютером. Дано описание функционала разработанной утилиты удаленного управления ПО, а также приведена диаграмма классов системы.

Список литературы

1. Полное руководство RedHat Package Manager. Режим доступа: http://uneex.ru/static/RedHatRPMGuideBook/rpm_guide-linux.html (дата обращения 23.05.2014).

2. Fedora Russian. Available at: http://wiki.russianfedora.pro/index.php?title=%D0%93%D0%BB%D0%B0%D0%B2%D0%BD%D0%B0%D1%8F, accessed 23.05.2014.
3. Галкин В.А., Григорьев Ю.А. Телекоммуникации и сети. М.: Издательство МГТУ им. Н.Э. Баумана, 2003. 609 с.
4. Павловская Т.А. C/C++. Программирование на языке высокого уровня. Спб.: Питер. 2009. 461 с.
5. YUM Package Manager. Режим доступа: http://yum.baseurl.org/ (дата обращения 22.05.2014).
6. Основы управления пакетами Debian. Режим доступа: http://www.debian.org/doc/manuals/debian-faq/ch-pkg_basics (дата обращения 22.05.2014).
7. Secure Shell. Режим доступа: http://en.wikipedia.org/wiki/Secure_Shell (дата обращения 22.05.2014).
8. Протокол SSH. Аутентификация по ключам. Режим доступа: http://vds-admin.ru/ssh/ssh-autentifikatsiya-po-klyucham-ispolzovanie-programm-ssh-keygen-i-ssh-agent (дата обращения 23.05.2014).