Национальный исследовательский
Томский государственный университет
Кемеровский государственный университет
Кемеровский научный центр СО РАН
Институт вычислительных технологий СО РАН
Филиал Кемеровского государственного университета
в г. Анжеро-Судженске

# ИНФОРМАЦИОННЫЕ ТЕХНОЛОГИИ И МАТЕМАТИЧЕСКОЕ МОДЕЛИРОВАНИЕ (ИТММ–2014)

**Материалы XIII Международной научно-практической конференции имени А. Ф. Терпугова**

**20–22 ноября 2014 г.**

**Часть 1**

Издательство Томского университета

2014

вычисления не окончены, возвращаемся к стеку, если вычисления окончены, агент выдает сообщение, содержащее результат вычислений, либо возвращает его назначению.

Необходимо отметить, что все агенты системы в процессе совершения действий отправляют сообщения в стек сообщений. Для представления агентов системы использованы потоки, в каждом из которых определяется объект класса, представляющий процесс системы. Атрибуты и методы класса отображают свойства действий, хранимых в онтологии. При этом такие компоненты модели, как поисковая система, декомпозитор, также являются агентами, выполняющими системные функции.

Таким образом, реализован один из подходов удаленного вычисления, в котором применение мультиагентных технологий позволяет реализовывать действия, прописанные в онтологии, давая возможность их применения в вычислениях либо циклах обработки информации. При этом использование многопоточного программирования позволяет значительно сократить время обработки информации в компьютерных сетях.

Литература

1. *Найханова Л. В.* Технология создания методов автоматического построения онтологий с применением генетического и автоматного программирования. – Улан-Удэ: Изд-во БНЦ СО РАН, 2008. – 244 с.

2. *Найханова Л. В*. О применении автоматного программирования в построении иерархии процессов в онтологии с активной семантикой / Л. В. Найханова, С. Д. Данилова, Н. Б. Ким // Теоретические и прикладные вопросы современных информационных технологий: матер. XXI Всерос. науч.-техн. конф. – Улан-Удэ: Изд-во ВСГУТУ, 2012. – С. 274–281.

## РАСПРЕДЕЛЁННАЯ СИСТЕМА СЕРВИСОВ

***Д. О. Змеев, О. А. Змеев, Д. А. Соколов, А. А. Цыганков***

*Национальный исследовательский*

*Томский государственный университет, Томск, Россия*

### Введение

В настоящее время становится все более актуальной задача информатизации различного рода организаций и автоматизации бизнес-процессов. Для небольших организаций эта задача может решаться созданием одного сайта, который будет выполнять роль визитной карточки, а также предоставлять некоторую функциональность, автоматизирующую работу с клиентами. Однако с увеличением размера организации и количества поставленных перед сайтом задач сложность этого сайта значительно возрастает, его разработка длится несколько итераций, а иногда вообще не рассчитана на завершение и направлена на постоянное совершенствование продукта. Кроме того, помимо задач внешнего направления (работа с потенциальными клиентами),

сайт должен решать задачи внутреннего направления, помогая сотрудникам организации выполнять их должностные обязанности, автоматизировать бизнес-процессы. Ключевой проблемой является величина разрабатываемой системы: поддержка и дальнейшая разработка будут стоить всё дороже и дороже.

Если посмотреть на существующие решения для крупных порталов, можно увидеть примеры Google, Yandex, MSN – все они строятся по принципу независимых сервисов (самостоятельных сайтов), которые общаются между собой посредством документированного API. Объединяет всё разнообразие сервисов в один портал сервис аккаунтов: каждый пользователь имеет свой аккаунт на портале, который используется равноправно на всех сервисах и который объединяет все данные о пользователе.

В данной работе авторы рассматривают некоторое общее описание системы распределённых сервисов, это сделано в качестве анализа подобных решений для сравнения с корпоративной образовательной социальной сетью, которая описана в [1].

**Описание решения**

Для реализации корпоративного портала авторами работы предлагается подобное решение: выбираются наиболее значимые потребности и под них разрабатываются сервисы. Каждый сервис работает независимо, имеет собственную базу данных и может быть написан на любом языке с использованием любых технологий: внутренняя архитектура сервиса остается вопросом команды, разрабатывающей этот сервис, и никоим образом не влияет на весь портал в целом.

**API и community**

Выше несколько раз была подчеркнута независимость сервисов, однако они не должны быть закрытыми. Взаимодействие между сервисами происходит посредством документированного API. Помимо взаимодействия между официальными частями портала, опубликованный API также позволяет сторонним разработчикам создавать свои приложения с использованием сервисов портала для разных целей под разные платформы, тем самым удобство пользования порталом повышается. К примеру, разработан сервис, который представляет собой календарь всех событий, которые происходят в организации. Естественно, календарь имеет некоторый API, чтобы другие сервисы (допустим, новостная лента) имели доступ к данным календаря. Сторонний разработчик, изучив опубликованный API, может написать Android-приложение, которое будет выводить все мероприятия организации либо только некоторые, необходимые по какому-то признаку.

**Зависимости**

В случае если один сервис использует данные другого сервиса с помощью его API, происходит зависимость одного сервиса от другого сервиса. Одной из задач является сделать наименьшую зависимость в портале, чтобы отдельные его части могли работать без других. Для решения этой

задачи каждый сервис должен изначально проектироваться из расчета, что некоторые сервисы необходимы для его работы, а с некоторыми он может взаимодействовать, но не обязан. Так, например, для работы сервиса новостной ленты необходим сервис аккаунтов, а также возможно взаимодействие с сервисом календаря событий, но можно обойтись и без него.

При изначальной настройке сервиса администратор должен ввести в соответствующие поля адреса взаимодействующих сервисов. Если адрес обязательного сервиса будет отсутствовать или по указанному адресу сервис не будет отвечать, то администратору выведется соответствующее предупреждение, а сам сервис не будет запускаться.

В случае если указаны адреса необязательных сервисов и они доступны, то при работе сервис будет предоставлять дополнительные возможности по интеграции.

**Узкие места**

При таком подходе некоторые сервисы могут стать узким местом всего портала. Например, скорее всего каждый сервис портала будет зависеть от сервиса аккаунтов и посылать к нему очень много запросов. Однако эта проблема решается тем, что сервисы достаточно небольшие и за их работу всегда отвечает отдельный человек, который точно знает его архитектуру, проблемные места и возможные пути решения. Для одного отдельного сервиса легко проводить аналитику и оптимизацию: перевести на нереляционную СУБД, создать структуру кеширования, оптимизировать запросы, переписать отдельные части или весь сервис на более быстрый при исполнении язык. При этом любые изменения не будут касаться других частей портала, могут проводиться сравнительно быстро, безопасно и своевременно.

**Масштабирование**

Корпоративный портал образовательного учреждения уровня университета – достаточно крупный продукт, требующий значительных вычислительных мощностей. Здесь встает вопрос масштабирования. Как известно, масштабирование бывает вертикальным и горизонтальным. Для простых (монолитных) систем легко применимо только вертикальное масштабирование, при этом горизонтальное масштабирование возможно, но разработка такого решения является очень трудоемкой, а следовательно, и дорогой. Однако наращивание мощности одного узла не может быть бесконечным, к тому же это также может быть дорого. Сложность узла возрастает, обслуживающий персонал должен быть более квалифицированным. Если узел выходит из строя, то выходит из строя весь портал. В случае использования множества независимых сервисов очень хорошо применимо горизонтальное масштабирование: каждый сервис может быть развернут на своем собственном узле. При этом при небольшой величине сервиса требования к узлу совсем небольшие: они просты, дёшевы, не требуются высококвалифицированные кадры. Если выйдет из строя один узел, выйдет

из строя только его сервис и, возможно, несколько зависимых сервисов, но не весь портал. Таким образом, надежность системы повышается.

**Внедрение**

Вопрос интеграции в таких системах стоит обычно очень остро по двум причинам:

1. Разработка такого большого продукта обходится очень дорого, и в итоге редко получается именно то, что хотели.

2. Организация пытается сверху навязать своим сотрудникам пользоваться разработанной системой, за которую заплачено много денег, но система может быть неудобна, а сотрудники консервативны.

Эти проблемы решаются тем, что отдельные небольшие сервисы портала разрабатываются последовательно, по мере необходимости, для конкретной целевой группы. На возникающие задачи в любом случае выделяются средства на разработку некоторого продукта, который её решит. Если эти средства потратить на разработку описанного сервиса, который будет решать возникшую задачу, но в то же время будет удовлетворять всем описанным условиям сервиса и будет доступен для других отделов, то постепенно вырастет сеть таких сервисов, которые образуют корпоративный портал организации. Тогда необходимость в обычных сайтах отделов отпадет, а пользователи будут постепенно привыкать к новым продуктам, причем в первую очередь ими будут пользоваться те, кто в них заинтересован, а не все подряд.

**Разнообразие, эксперименты**

Каждый сервис пишется и исполняется независимо, а общение происходит с помощью API, это позволяет каждый сервис писать на любом удобном языке с использованием любых удобных технологий. На данный момент в IT-индустрии самым дорогим ресурсом являются квалифицированные кадры. Набрать большое количество квалифицированных людей со знанием одного набора языков и технологий зачастую является проблемой. В случае небольших независимых сервисов каждая команда сама решает, на каком языке он будет написан и какие технологии будут использоваться. При этом решение может основываться как на особенностях и потребностях самого сервиса, так и на знаниях и умениях разработчиков.

Помимо прочего, появляется возможность для экспериментов: чтобы протестировать какую-то новую технологию, язык или другие средства разработки, достаточно применить их на одном незначительном сервисе или написать новый, при этом остальная часть портала никак не пострадает. Если эксперимент будет удачным, то полученный опыт можно будет постепенно перенести на другие сервисы.

При этом, конечно, стоит избегать слишком большого разнообразия в используемых средствах: каждый участник команды должен иногда работать в другой команде, чтобы производить обмен опытом и создавать единообразный продукт. Если развести слишком большой «парк» различных средств, то рано или поздно он выйдет из-под контроля и начнет развали-

ваться: не будет людей, которые бы понимали всю структуру портала, а разные команды не будут понимать друг друга. К тому же всегда необходимо помнить, что любой разработчик может в любой момент уйти, и его необходимо будет вовремя заменить. В связи с этим стоит жестко зафиксировать принципы работы и следить за их исполнением, соблюдая во всех сервисах однообразие во всём, включая стиль написания кода, применяемые технологии, внешний вид, принципы администрирования.

**Общий код**

Очевидно, что разные сервисы, хоть и решают разные задачи, но также очень часто возникают одинаковые задачи или какие-то внутренние нужды. Можно каждый раз писать код заново, который будет решать такие задачи. Можно копировать из проекта в проект, при этом если код меняется в одном сервисе, желательно его изменить во всех остальных. Всё это очень усложняет разработку. Но можно сформировать общий банк библиотек, в котором всегда будет свежая версия. Основную часть портала планируется разрабатывать на ASP.NET MVC. В связи с этим представляется очень удобным использовать менеджер пакетов NuGet, который позволяет развернуть собственный корпоративный банк библиотек, куда разработчики могут добавлять свои библиотеки или скачивать библиотеки, разработанные коллегами. При этом установка, контроль за версией пакета, а также разрешение зависимостей происходят автоматически, что очень сильно упрощает работу разработчиков и повышает надежность продукта.

**Заменимость**

Данным продуктом не стоит вытеснить существующие. Продукт предназначен для предоставления функционала, которого нет сейчас в доступе, или есть, но с критичными недостатками. Сервисы могут быть тесно интегрированы с другими предоставляемыми в Интернете продуктами. Например, для хранения файлов можно использовать Google Drive: для многих качества и функциональности этого продукта достаточно. Однако в некоторых случаях нежелательно использовать сторонние ресурсы для хранения информации. При решении отдельных задач одним из требований является закрытость информации, в этих условиях невозможно использовать услуги сторонних организаций, и единственное решение – разрабатывать собственный сервис, который бы максимально удовлетворял потребностям пользователя портала.

Таким образом, отдельные функции портала могут реализовывать как собственные сервисы, так и сторонние продукты, взаимодействие с которыми также будет производиться через API.