OSTIS-2015
(Open Semantic Technologies for Intelligent Systems)

УДК 004.822:514

РЕАЛИЗАЦИЯ ПЛАТФОРМЫ ДЛЯ WEB-ОРИЕНТИРОВАННЫХ СИСТЕМ, УПРАВЛЯЕМЫХ ЗНАНИЯМИ

Корончик Д. Н.

[*] *Белорусский государственный университет информатики и радиоэлектроники, г. Минск, Республика Беларусь*

denis.koronchik@gmail.com

В работе приводится описание технической реализации платформы для web-ориентированных систем, управляемых знаниями. Описаны основные принципы, которые лежат в основе платформы: ориентация на коллективную разработку базы знаний, возможность отображения (редактирования) базы знаний на различных внешних языках и их отображение в режиме реального времени.
Ключевые слова: клиент-сервер, компоненты пользовательского интерфейса, платформа.

Введение

В рамках *Проекта OSTIS* [**Голенков, 2013**] ведется разработка платформы, которая позволяет реализовывать web-ориентированные системы, управляемые знаниями. Познакомиться с описанием предыдущей версией можно по ссылке **[Корончик, 2014]**. Основные положения, которые лежат в основе указанной платформы следующие:

- база знаний (БЗ) системы редактируется множеством разработчиков, что можно даже делать во время эксплуатации системы;
- разграничение доступа к различным частям базы знаний;
- возможность отображать (редактировать) части базы знаний с использованием различных внешних языков (SCg-код [**Голенков и др, 2001**], SCn-код, чертежи, ЕЯ и т. д.);
- изменения в базе знаний в режиме реального времени должны отображаться у всех пользователей системы.

Архитектура

Архитектура разработанной платформы показана на рисунке 1. Используется клиент-серверная архитектура, где клиентская реализована с использованием web-технологий. Использование web-технологий в реализации клиентской части позволяет использовать приложение в любом (достаточно новом) браузере. Как видно из рисунка 1, платформа состоит из следующих частей:

- http сервер – серверное приложение, которое выполняет следующие функции:
  - обработка http запросов (их список можно посмотреть в **[sc-web api, 2014]**);
  - перенаправления запросов из клиентской части в sc-память и событий из sc-памяти в клиентскую часть;
  - авторизация и уровни доступа пользователей;
- sc-память – это хранилище текстов записанных с помощью SC-кода **[Корончик, 2013]**;
- машина обработки знаний – набор sc-агентов, которые обеспечивают функционирование системы;
- компоненты ядра пользовательского интерфейса (ПИ) – это основа клиентской части, которая обеспечивает интеграцию компонентов ПИ и их взаимодействие с базой знаний;
- компоненты ПИ – обеспечивают ввод и вывод информации представленной с помощью различных внешних языков.

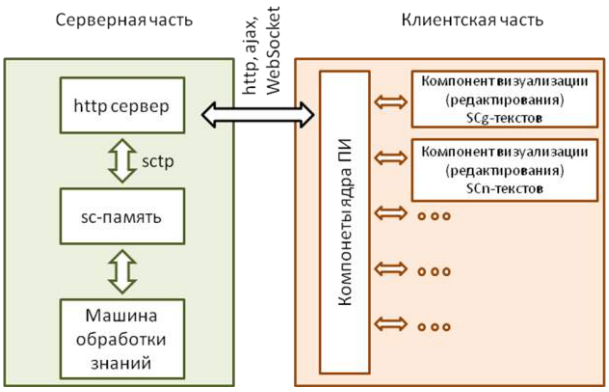

Рисунок 1 – Архитектура программной платформы для реализации web-ориентированных систем основанных на знаниях

Реализация **http сервера** основана на использовании библиотеки tornado **[tornado, 2014]**. Предыдущая версия http сервера была реализована с помощью библиотеки django. Однако для того, чтобы появилась возможность реализации sc-агентов ПИ на клиентской части, необходимо было реализовать возможность инициировать функции в JavaScript по событиям из sc-памяти. С этой целью было принято решение использовать стандарт WebSocket **[websocket, 2014]**. В связи с этим была использована именно библиотека tornado.

При старте приложения в браузере, устанавливается WebSocket соединение между сервером и клиентской частью. На стороне клиента реализована библиотека, которая позволяет, используя бинарный протокол sctp **[sctp, 2014]**, работать с sc-памятью, для этого в http сервере имеется небольшой модуль, который преобразует запросы из WebSocket в TCP/IP Socket и обратно (параллельно проверяя наличие необходимого уровня доступа на выполнение той или иной команды). Авторизация на сервере реализована с помощью google (пользователи имеющие акант google могут авторизоваться на сайте). Каждому пользователю устанавливается роль (по умолчанию это Гость с минимальными правами на чтение и редактирование). Управление ролями пользователя может выполнять администратор через панель администратора (как и во всех современных сайтах).

Чтобы обеспечить возможность одновременной работы множества пользователей с системой была пересмотрена и значительно улучшена реализация **sc-памяти [Корончик, 2013]**. Основные изменения, которые были внесены в реализацию sc-памяти:

- переработан механизм синхронизации при обработке команд из разных потоков. Раньше выполнение команды одного потока блокировало выполнение команд из других потоков, что приводило к большому падению производительности при увеличении числа потоков (клиентов). Новый механизм делит каждый сегмент на N частей. При попытке блокировать элемент со смещением X в сегменте, он блокирует X % N часть (% - операция получения остатка от деления). Таким образом, при обращении к элементам в разных сегментах или разных частях сегмента, выполняемые в разных потоках команды не блокируют друг друга;
- уменьшено количество потребляемой оперативной памяти, за счет уменьшения размера ячейки sc-памяти с 56 до 44 байт, что дает экономию в 21%. Это было достигнуто за счет удаления временных меток из sc-элемента. Теперь команда удаления сразу удаляет элементы, чтобы остальные команды не могут использовать удаляемые элементы, так как они блокируются. Если на удаляемый sc-элемент указывает итератор, то элемент просто помечается как необходимый к удалению (он становится недоступным для инициированных после этого команд) и как только количество ссылающихся на него итераторов становится равным нулю, элемент удаляется;
- для каждого sc-элемента были добавлены поля, которые обозначают уровень доступа к sc-элементу. Выделено 16 уровней доступа на чтение и 16 уровней доступа на запись (соответственно для хранения используется 1 байт). Правило работы с ними очень простое – каждой программе (sc-агенту, пользователю), которая использует sc-память, ставится в соответствие её уровень доступа. Эта программа имеет право изменять (читать) sc-элемент, если её уровень доступа на изменение (чтение) больше, либо равен уровню доступа у sc-элемента. Кроме того, если уровень доступа позволяет изменять sc-элемент, то программа имеет право изменить его на любое другое значение (как в меньшую, так и в большую сторону).

Машина обработки знаний не претерпела больших изменений. Она была лишь адаптирована под изменения, сделанные в реализации sc-памяти. Сама по себе МОЗ на текущий момент представляет собой набор библиотек расширений, которые загружаются при старте sc-памяти и регистрируют в ней обработчики на события (sc-агенты).

Компоненты ядра пользовательского интерфейса реализованы на клиентской части приложения с использованием JavaScript. По сути, они отвечают за весь базовый функционал клиентской части приложения и обеспечивают корректную работу компонентов. Вкратце опишем функции, которые выполняют компоненты ядра ПИ:

- отображение и инициирование команд ПИ, которые располагаются в главном меню **[Корончик2, 2013]**;
- логика работы с аргументами команд. В качестве аргумента команды может быть указан любой элемент на странице, у которого установлен атрибут *sc_addr* (адрес sc-элемента в памяти);
- навигация по истории диалога пользователя с системой;
- переключение между режимами идентификации (русский, английский). Ядро автоматически запрашивает идентификаторы на необходимом языке и меняет их для всех элементов на странице с атрибутом *sc_addr*. Если же идентификатор для указанного языка не найден, то устанавливается системный идентификатор sc-элемента;
- обеспечивает взаимодействие между компонентами пользовательского интерфейса. Именно ядро принимает решение, какой компонент использовать для отображения информации в том или ином виде. Пользователь лишь указывает внешний язык, на котором он хочет увидеть ответ на свой запрос. Взаимодействие между компонентом и ядром ПИ осуществляется через песочницу - component sandbox. Компонент в своей работе может работать лишь со своей песочницей, не используя функции ядра напрямую. Это позволяет значительно сократить временные затраты на поддержку большого числа

компонентов, когда происходят изменения в ядре ПИ;

- обеспечивает поиск по идентификатору. При введении пользователем 3-х или более символов в строку поиска, ядро делает запрос на сервер, где происходит поиск всех идентификаторов, которые содержат указанную подстроку символов. На сервере имеется sc-агент, который реагирует на появление новых идентификаторов в системе, и добавляет их в базу ключ значений redis (в которой их ищет сервер по запросу клиента);
- отображение всплывающих подсказок. Для каждого элемента на странице с атрибутом sc_addr, при наведении, формируется всплывающая подсказка. Эта подсказка описана в базе знаний для разных языков (рисунок 2)

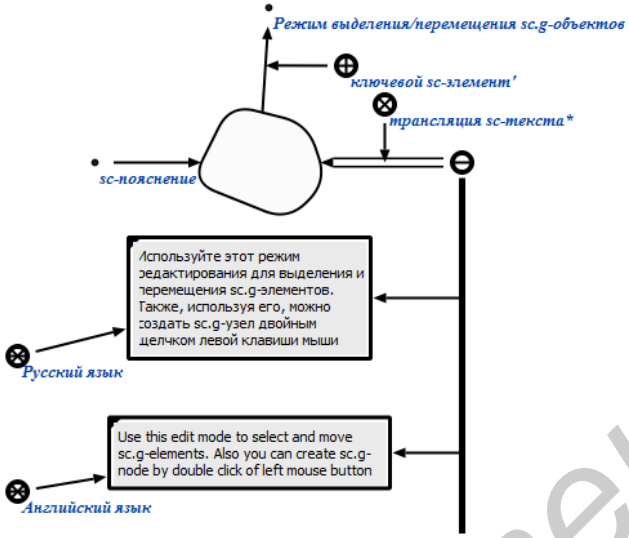

Рисунок 2 – пример указания пояснений для элемента управления в редакторе sc.g-текстов

Компоненты пользовательского интерфейса реализуются также с использованием JavaScript. Раньше мы рассматривали лишь один вид компонентов – компоненты, которые позволяют просматривать (редактировать) содержимое sc-ссылок (файлы). Но для того чтобы изменения в базе знаний отображались у пользователей в режиме реального времени этого было недостаточно. Чтобы решить данную проблему были добавлены компоненты, которые позволяют просматривать (редактировать) структуры напрямую из базы знаний.

При регистрации в ядре, компоненту ставится в соответствие некоторое его описание, где указано, может ли он отображать (редактировать) содержимое sc-ссылок и/или структур. В JavaScript коде это выглядит следующим образом:

```
SCgComponent = {
    ext_lang: 'scg_code',
    formats: ['format_scg_json'],
    struct_support: true,
    factory: function(sandbox) {
        return new scgWindow(sandbox);
    }
};
```

Рассмотрим поля объекта *SCgComponent* подробнее:

- *ext_lang* – в этом поле указывается системный идентификатор sc-узла, который обозначает внешний язык с которым работает компонент;
- *fomats* – в этом поле указывается список системных идентификаторов sc-узлов, которые обозначают форматы файлов (sc-ссылок) поддерживаемые компонентом. Если данное поле пусто, то компонент не может отображать (редактировать) sc-ссылки;
- *sturct_support* – наличие этого поля со значением true, указывает на то, что данный компонент умеет отображать (редактировать) структуры напрямую из памяти;
- *factory* - это функция, которая вызывается, когда ядру ПИ необходимо создать экземпляр данного компонента. В качестве параметра передается экземпляр объекта component sandbox, через который будет осуществляться взаимодействие между компонентом и ядром. Возвращает функция экземпляр созданного компонента.

В момент создания экземпляра компонента доступна следующая информация:

- *sandbox.container* – это строка, которая содержит id элемента на странице, куда должен быть встроен компонент;
- *sandbox.canEdit()* – функция, которая возвращает true, если требуется создать редактор, а не просмотрщик;
- *sandbox.is_struct* – поле, которое равно true, если необходимо просмтаривать (редактировать) структуры, а не sc-ссылки;
- *sandbox.addr* – это адрес структуры или sc-ссылки с которой работает компонент.

В момент создания компонент должен подписаться на необходимые ему события (установить в соответствующие поля sandbox функции обработчики):

- *sandbox.eventDataAppend* - это событие, которое вызывается, когда необходимо отобразить содержимое sc-ссылки. В качестве аргумента функция обработчика принимает данные (содержимое sc-ссылки);
- *sandbox.getObjectsToTranslate* – данное событие вызывается ядром, когда происходит переключение между режимами идентификации. Обработчик данного события – это функция, которая возвращает список всех sc-адресов отображаемых (редактируемых) элементов;
- *sandbox.eventApplyTranslation* – это событие, которое происходит, когда необходимые идентификаторы для нового режима идентификации найдены. Обработчик данного события принимает в качестве параметра объект, полями которого являются адреса sc-элементов, а значениями – новые идентификаторы. Если же для

какого-то элемента нет поля, то идентификатор для него не найден;

- *sandbox.eventStructUpdate* – это событие инициируется, когда изменяется отображаемая структура (появляются или удаляются выходящие дуги из sc-узла обозначающего эту структуру).

Таким образом, если нам необходимо реализовать компонент, который отображает содержимое sc-ссылок, необходимо:

- описать формат данных в базе знаний;
- зарегистрировать компонент для отображения нового формата;
- реализовать функцию *sandbox.eventDataAppend*, которая на вход примет данные из sc-ссылки. Эти данные уже необходимо обрабатывать в компоненте и как-то визуализировать.

Заключение

В настоящее время исходные коды серверной и клиентской частей можно найти по ссылке **[sc-web, 2014]** там же во вкладке “Issues” находится список текущих задач. Инструкцию по установке можно найти там же на wiki (страница “version dev”), данная инструкция постоянно обновляется. Исходные коды реализации sc-памяти можно найти по ссылке **[sc-memory, 2014]**. Текущие задачи по ней находятся также во вкладке “Issues”.

Хотелось бы отметить, что еще не все задачи решены, но уже на текущем этапе можно делать системы, управляемые знаниями, с возможностью коллективной разработки базы знаний и градацией разработчиков и пользователей по уровням доступа и не только через web-интерфейсы. При этом одну и ту же информацию пользователи смогут редактировать одновременно с использованием различных внешних представлений, которые синхронизируются в режиме реального времени (альтернативы этому найти в интернете не удалось). Сейчас эти наработки используются в метасистеме IMS OSTIS **[ims ostis, 214]**.

В ближайшее время планируются следующие доработки в платформе:

- улучшение sctp протокола за счет добавления более сложных запросов (создать целую конструкцию по шаблону, создать N узлов и т. д.), что позволит значительно ускорить взаимодействие клиентской части с sc-памятью;
- создание полноценной библиотеки компонентов ПИ с возможность установки в новую систему по сети;
- разработка распределенной версии sc-памяти.

Библиографический список

[Голенков и др, 2001] Представление и обработка знаний в графодинамических ассоциативных машинах /В. В. Голенков, [и др]; – Мн. : БГУИР, 2001

[Голенков, 2013] Открытый проект, направленный на создание технологии компонентного проектирования интеллектуальных систем / В. В. Голенков, Н. А. Гулякина // Материалы международной научно-технической конференции «Открытые семантические технологии проектирования интеллектуальных систем» - Минск, 2013

[Корончик, 2013] Реализация хранилища унифицированных семантических сетей / Д. Н. Корончик // Материалы международной научно-технической конференции «Открытые семантические технологии проектирования интеллектуальных систем» - Минск, 2013

[Корончик, 2014] Пользовательский интерфейс интеллектуальной метасистемы поддержки проектирования интеллектуальных систем / Д. Н. Корончик // Материалы международной научно-технической конференции «Открытые семантические технологии проектирования интеллектуальных систем» - Минск, 2014

[Корончик2, 2013] Унифицированные семантические модели пользовательских интерфейсов интеллектуальных систем и технология их проектирования / Д. Н. Корончик // Материалы международной научно-технической конференции «Открытые семантические технологии проектирования интеллектуальных систем» - Минск, 2013

[ims ostis, 2014] Сайт метасистемы IMS OSTIS [Электронный ресурс]. – 2014 – Режим доступа: http://ims.ostis.net/ – Дата доступа: 02.11.2014

[sctp, 2014] Протокол sctp [Электронный ресурс]. – 2014 – Режим доступа: https://github.com/deniskoronchik/sc-machine/wiki/sctp – Дата доступа: 02.11.2014

[sc-memory, 2014] Исходные коды реализации sc-памяти [Электронный ресурс]. – 2014 – Режим доступа: https://github.com/deniskoronchik/sc-machine – Дата доступа: 05.11.2014

[sc-web, 2014] Исходные коды реализации web-платформы [Электронный ресурс]. – 2014 – Режим доступа: https://github.com/deniskoronchik/sc-web – Дата доступа: 05.11.2014

[sc-web api, 2014] Rest api платформы [Электронный ресурс]. – 2014 – Режим доступа: https://github.com/deniskoronchik/sc-web/wiki/Api – Дата доступа: 03.11.2014

[tornado, 2014] Tornado Web Server [Электронный ресурс]. – 2014 – Режим доступа: http://www.tornadoweb.org/en/stable/ – Дата доступа: 05.11.2014

[websocket, 2014] WebSocket [Электронный ресурс]. – 2014 – Режим доступа: https://www.websocket.org/ – Дата доступа: 05.11.2014

IMPLEMENTATION OF WEB-PLATFORM FOR SYSTEMS BASED ON KNOWLEDGES

Koronchik D. N

**Belarusian State University of Informatics and Radioelectronics, Minsk, Republic of Belarus*

denis.koronchik@gmail.com

This article describes tehcnical aspects web-platform for systems based on knowlegdes. Described platform allows developers to edit knowledge base of system by using different languages and instruments with real-time synchronization. Another words if one developer describes solar system in any formal language, and another developer draw it in special tool, they will have synchronized their views of data by system.