OSTIS-2014
(Open Semantic Technologies for Intelligent Systems)

УДК 004.822:514

ПОЛЬЗОВАТЕЛЬСКИЙ ИНТЕРФЕЙС ИНТЕЛЛЕКТУАЛЬНОЙ МЕТАСИСТЕМЫ ПОДДЕРЖКИ ПРОЕКТИРОВАНИЯ ИНТЕЛЛЕКТУАЛЬНЫХ СИСТЕМ

Корончик Д. Н.

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

denis.koronchik@gmail.com

В работе описывается реализованный пользовательский интерфейс для интеллектуальной метасистемы поддержки проектирования интеллектуальных систем, которая располагается по адресу http://ims.ostis.net. В рамках статьи рассмотрены команды доступные пользователю в рамках ПИ, компоненты, которые реализованы для его функционирования и основные идеи, которые лежат в его основе.
Ключевые слова: пользовательский интерфейс; компонент пользовательского интерфейса; команда пользовательского интерфейса.

Введение

В рамках *Проекта OSTIS* [Голенков, 2013] ведется разработка ***SC-технологии проектирования пользовательских интерфейсов*** (семантическая технология проектирования пользовательских интерфейсов интеллектуальных систем). Более подробно с её описанием можно ознакомиться по ссылке [Корончик, 2012]. Основные положения, которые лежат в основе указанной технологии следующие:

- *пользовательский интерфейс* (ПИ) рассматривается как специализированная интеллектуальная система, которая направлена на получение сообщений от пользователя и вывода ему ответов системы. Следовательно, пользовательский интерфейс, как и любая другая система, разрабатывается по *Технологии OSTIS* является многоагентной системой, основанной на знаниях и, прежде всего, на онтологиях. Основной задачей пользовательского интерфейса является перевод сообщения от пользователя, полученного на некотором внешнем языке, на внутренний язык системы (SC-код), а также перевод ответа системы на некоторый внешний язык, понятный пользователю и отображение этого ответа;
- в основе графических интерфейсов лежит SCg-код (Semantic Code graphical – который является одним из возможных способов визуального представления текстов SC-кода) [Голенков и др, 2001]. Объекты, отображаемые на экране с помощью SCg-кода, будем называть sc.g-элементами. Основным принципом, положенным в его основу, является то, что все изображенные на экране объекты (sc.g-элементы), в том числе и элементы управления, являются изображением узлов семантической сети. Другими словами каждому изображенному на экране объекту соответствует узел в семантической сети (базе знаний) синонимичный ему;
- формализация семантики пользовательских действий, с последующим анализом, а также их унификация и четкая типология.

В данной статье рассматривается пользовательский интерфейс интеллектуальной метасистемы поддержки проектирования интеллектуальных систем (*Метасистемы IMS.OSTIS*) реализованной по *Технологии OSTIS* **[OSTIS, 2013]**.

1. Внешний вид пользовательского интерфейса

Внешний вид главного окна пользовательского интерфейса *Метасистемы IMS.OSTIS* представлен на рисунке 1. В основе этого ПИ лежит *SCg-код*. Все объекты (в том числе и элементы управления) изображаемые в рамках главного окна, являются знаками некоторых sc-элементов в базе знаний. Это позволят пользователю указывать их в качестве аргументов для различных команд.

Рисунок 1 - Внешний вид пользовательского интерфейса метасистемы

2. Взаимодействие с пользователем

Взаимодействие пользователя с системой осуществляется через обмен сообщениями. Сообщения пользователь формирует с использованием пользовательских команд. Команды пользователя инициируются с помощью заранее определенной последовательности элементарных пользовательских действий. *Элементарное пользовательское действие* - это действие, которое осуществляется с использованием некоторого устройства ввода и не может быть декомпозировано (разбито на другие, более мелкие действия). Примерами таких действий могут быть, нажатия (отпускания) клавиш мыши или клавиатуры, перемещение мыши и т. д. Выделены следующие классы команд пользователя по типу инициируемого действия:

- *команда-вопрос* - класс команд, которые инициируют вопрос системе;
- *команды редактирования* - класс команд, которые инициируют действия связанные с редактированием внешних *sc-текстов*;
- *команды просмотра* - класс команд, которые инициируют действия связанные с просмотром внешних *sc-текстов*.

По способу определения аргументов выделены следующие классы команд пользователя:

- класс команд с заранее определенными аргументами, т. е., объекты, над которыми инициируется действие, заранее известны. Например, команда раскрытия пункта меню - это команда вывода декомпозиции объекта, где этим объектом является она сама;
- класс команд с дополнительно указываемыми аргументами. В этих командах, объекты, над которыми инициируется действие, необходимо указать.

Существует базовый способ инициирования команд с дополнительно указываемыми аргументами. Сначала указываются аргументы команды. Они просто перетягиваются на панель аргументов. После чего инициируется команда, щелчком левой клавиши мыши на знаке соответствующего класса команд.

Команды с заранее определенными аргументами инициируются щелчком левой клавиши мыши на знаке, обозначающем эту команду.

В системе, к настоящему моменту, реализованы следующие *пользовательские команды*:

- запрос полной семантической окрестности. Данная команда может быть инициирована двумя способами. Первый способ: указать как аргумент *sc-элемент* к которому задается вопрос; инициировать команду левым щелчком мышки на её знаке в главном меню (*Просмотр БЗ > Семантическая окрестность > Полная семантическая окрестность*). Второй способ: инициировать команду с заранее определенными аргументами - это ссылки в рамках страницы. Инициирование также происходит по щелчку левой клавиши мыши по знаку команды (курсивный текст синего цвета, который подчеркивается при наведении курсора мыши);
- команда переключения активного окна. Эта команда инициируется с помощью щелчка левой клавиши мыши на знаке окна (в левой части главного окна). После того, как окно становится активным, знак его обозначающий становится синего цвета, а его содержимое отображается в центральной области главного окна;
- команда изменения режима идентификации. Чтобы инициировать эту команду необходимо сделать щелчок левой клавиши мыши на кнопку с треугольником, которая расположена справа от текущего режима идентификации. Затем, в появившемся списке, сделать щелчок левой клавиши мыши по одному из доступных режимов идентификации;
- команда подготовки содержимого активного окна к печати. Она инициируется нажатием левой клавиши мыши на команде с изображением принтера, которая расположена над списком окон, в левой части главного окна. После инициирования команды в браузере создается вкладка, содержимое которой можно выводить на печать с помощью браузера.

3. Логико-семантическая модель пользовательского интерфейса

Как и любая система, построенная с использованием *Технологии OSTIS*, пользовательский интерфейс метасистемы строится с использованием компонентного подхода. Выделены следующие классы компонентов ПИ:

- *компоненты трансляции*. Компоненты данного класса обеспечивают трансляцию из *SC-кода* на внешний язык и обратно;
- *компоненты визуализации*. Компоненты данного класса обеспечивают вывод информации представленной на внешнем языке;

- *компоненты редактирования*. Компоненты данного класса обеспечивают ввод информации пользователем на внешнем языке.

Каждый компонент состоит из некоторого фрагмента базы знаний и набора sc-агентов. Из компонентов трансляции в ПИ метасистемы реализованы следующие компоненты:

- транслятор из *SC-кода* в *формат SCs Json* (формат схожий по структуре с *SCs-кодом 1-го уровня*, адаптированный для Web);
- транслятор из *формата SCs Json* в *SC-код*;
- транслятор *SCs-кода* в *SC-код*;
- транслятор *SCg-кода* в *SC-код*;
- транслятор из *SC-кода* в *SCg-код*.

Кроме компонентов трансляции в ПИ метасистемы реализованы следующие компоненты визуализации:

- визуализатор *sc.s-текстов* и *sc.n-текстов* [Голенков, 2012];
- визуализатор *sc.g-текстов*;
- визуализатор гипермедийных текстов (html);
- визуализатор карт (Google Maps);
- визуализатор видео (youtube).

Вся ***база знаний пользовательского интерфейса*** метасистемы делится на следующие разделы:

- описание пользователей – это часть базы знаний, в которой хранится информация обо всех пользователях системы (предпочитаемый естественный язык, различные настройки, протоколы действия пользователей и т. д.);
- описание синтаксиса и семантики всех используемых внешних языков – это часть базы знаний, в которой хранится формальное описание всех внешних языков представления знаний, которые используются при диалоге с пользователем. Примерами таких описаний могут быть: описание *SCg-кода*, *SCs-кода* и *SCn-кода*;
- описание используемых компонентов – это часть базы знаний, в которой описаны все компоненты пользовательского интерфейса, используемые в системе;
- описание *пользовательских команд* – это часть базы знаний, в которой хранятся описания команд, с помощью которых пользователь может взаимодействовать с системой;
- описание принципов работы самого пользовательского интерфейса – часть базы знаний, в которой хранится вся справочная информация по пользовательскому интерфейсу.

Абстрактная машина обработки знаний пользовательского интерфейса состоит из следующих *sc-агентов* (помимо sc-агентов, которые являются частью компонентов трансляции и отображения):

- *sc-агент генерации экземпляра команды, по её обобщенному описанию*. Все команды пользовательского интерфейса представлены в базе знаний и имеют некоторое обобщенное описание (шаблон), на основании которого данный *sc-агент* генерирует экземпляр этой команды с подставленными в неё аргументами;
- *sc-агент инициирования трансляции ответов на пользовательские запросы*. Данный sc-агент ожидает ответ на запрос пользователя и генерирует экземпляр команды, которая инициирует трансляцию ответа на внешний язык;
- *sc-агент сборки мусора*. Данный sc-агент осуществляет поиск и удаление информации, которая уже не актуальна. Такой информацией может быть уже завершенная команда трансляции и т. д.;
- *sc-агент сбора sc-идентификаторов*. Данный sc-агент занимается сбором всех идентифицируемых sc-элементов, для того, чтобы обеспечить поиск элементов по идентификатору.

4. Архитектура технической реализации

На рисунке 2 представлена архитектура реализованного пользовательского интерфейса. Машина обработки знаний ПИ располагается на серверной стороне. Клиентская часть лишь делает запросы с помощью http, ajax и получает на них ответы. При получении ответа изменяется состояние на клиентской части.

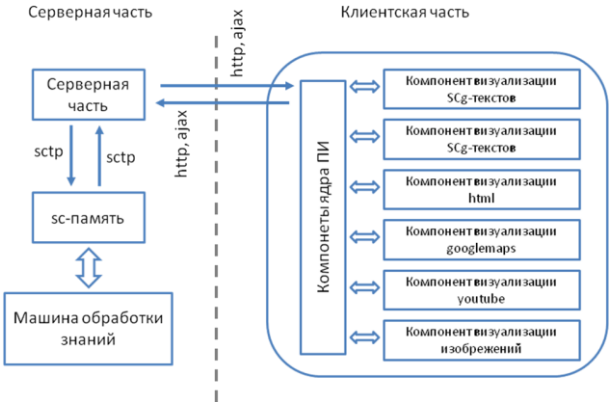

Рисунок 2 - Архитектура реализованного пользовательского интерфейса

Серверная часть ПИ реализована на языке python, с использованием django [django, 2013] она взаимодействует с sc-памятью с использованием протокола sctp [sctp, 2013]. *SC-агенты* пользовательского интерфейса (включая *sc-агенты* трансляции) реализованы на *языке программирования* Си. Программная реализация *SC-памяти* также реализована на *языке программирования* Си. Описание её архитектуры можно найти по ссылке [Корончик, 2013].

Клиентская часть ПИ состоит из компонентов ядра, которые обеспечивают взаимодействие с сервером, и компонентами, которые решают задачу визуализации и редактирования информации на внешних языках.

Заключение

Описанный в данной статье пользовательский интерфейс реализован и используется в рамках интеллектуальной метасистемы поддержки проектирования интеллектуальных систем. Посмотреть её можно на сайте http://ims.ostis.net. Можно выделить следующие преимущества пользовательского интерфейса этой системы:

- к элементам управления можно задавать любые вопросы. Это значительно сокращает стартовую инструкцию по использованию системы, так как достаточно лишь научить пользователя задавать вопросы базовым способом (указывая аргументы), а дальше он может узнать всю необходимую информацию по различным командам и элементам управления, задавая к ним вопросы;
- документация (help) по пользовательскому интерфейсу является частью самого ПИ, что сокращает накладные расходы по его разработке. В современных подходах необходимо реализовать ПИ, а затем еще и написать help по нему, который, при изменении самого ПИ, необходимо актуализировать. Здесь же описание команд используется самим пользовательским интерфейсом для функционирования;
- интернационализация. Простая возможность перехода между различными естественными языками. Используемый подход идентификации значительно позволяет сократить проблемы с переводимой информации. К примеру, в современной wikipedia, каждой статье в соответствие ставятся статьи на других языках. При этом они могут сильно отличаться друг от друга по содержанию и стилю, а также многие из них являются более актуальными, чем другие. В данном ПИ меняются лишь идентификаторы sc-узлов в БЗ, а сама информация остается неизменной, что сводит проблему интернационализации к добавлению новых идентификаторов и трансляция SC-текстов на необходимом языке;
- компонентный подход. Благодаря компонентному подходу можно легко расширять функционал ПИ системы, за счет добавления новых компонентов;
- многообразие отображаемых видов знаний. С помощью набора компонентов для отображения информации на различных языках, система может отображать любые знания. При этом имеется ряд компонентов, которые позволяют визуализировать любые знания с помощью унифицированных языков представления знаний: *SCg-код*, *SCn-код*, *SCs-код*;

Библиографический список

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

[Голенков, 2012] Графодинамические модели параллельной обработки знаний / В. В. Голенков, Н. А. Гулякина // Материалы международной научно-технической конференции «Открытые семантические технологии проектирования интеллектуальных систем» - Минск, 2012

[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, 2012] Семантические модели мультимодальных пользовательских интерфейсов и семантическая технология их проектирования / Д. Н. Корончик // Материалы международной научно-технической конференции «Открытые семантические технологии проектирования интеллектуальных систем» - Минск, 2012

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

[django, 2013] The Web framework for perfectionists with deadlines [Электронный ресурс]. – 2013 – Режим доступа: https://www.djangoproject.com – Дата доступа: 02.10.2013

[OSTIS, 2013] Открытая семантическая технология проектирования интеллектуальных систем [Электронный ресурс]. – 2013. - Режим доступа: http://ostis.net. – Дата доступа: 11.10.2013

[sctp, 2013] Протокол sctp [Электронный ресурс]. – 2013 – Режим доступа: https://github.com/deniskoronchik/sc-machine/wiki/sctp – Дата доступа: 02.10.2013

THE USER INTERFACE OF THE INTELLIGENT DESIGN SUPPORT METASYSTEM FOR INTELLIGENT SYSTEMS

Koronchik D. N.

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

denis.koronchik@gmail.com

This article describes implementation of semantic user interface for intelligent metasystem that provides support of intelligent system development.

Introduction

User interface is an intelligent system that works to provide dialog between user and system. Like another intelligent systems developed with OSTIS technology, it consist of knowledge base and knowledge processing machine.

Main Part

Knowledge base of UI for metasystem consist of next sections: user profiles; description of syntax and semantic of used external languages; description of used UI components; description of UI commands; description of principles that used in UI.

Knowledge process machine consist of next sc-agents: sc-agent to generate instance of command based on template; sc-agent to initiate translation of information to SC-code and backward; sc-agent to collect garbage; sc-agent to collect sc-identifiers.

Conclusion

Implemented UI has many advantages. It allows used to ask any questions about UI controls. It's simplify internationalization problems. Also it can be simply extended by development of new components.