OSTIS-2011
(Open Semantic Technologies for Intelligent Systems)

**УДК 004.822:514**

# СЕМАНТИЧЕСКАЯ ТЕХНОЛОГИЯ КОМПОНЕНТНОГО ПРОЕКТИРОВАНИЯ ПОЛЬЗОВАТЕЛЬСКИХ ИНТЕРФЕЙСОВ ИНТЕЛЛЕКТУАЛЬНЫХ СИСТЕМ

Д.Н. Корончик (*denis.koronchik@gmail.com*)

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

В работе приводится описание технологии проектирования пользовательских интерфейсов для интеллектуальных систем, основанных на семантических сетях. Результаты, описанные в данной работе, являются частью международного открытого проекта OSTIS (Open Semantic Technologies for Intelligent Systems). Использование предложенной технологии позволяет сократить сроки проектирования, повысить качество проектируемых интерфейсов, а также снизить требования к начальной квалификации разработчика.

*Ключевые слова:* визуализация знаний, компонентное проектирование, пользовательский интерфейс, технология, унифицированная модель

## Введение

Эффективность использования программной системы зависит от ее пользовательского интерфейса. В большинстве случаев разработка пользовательского интерфейса в современных системах отнимает большую часть времени затрачиваемого на разработку всей системы [Myers B.A., 1992]. Трудоемкость разработки обусловлена не столько сложностью пользовательского интерфейса, сколько отсутствием хорошо продуманных технологий их проектирования. Проектирование пользовательских интерфейсов интеллектуальных систем является более сложной задачей по ряду причин, что делает разработку технологии для их проектирования более актуальной. Сложность разработки таких интерфейсов обусловлена требованиями, которые предъявляются к ним:

- должны отображать различные виды знаний (при прочих равных условиях, чем больше различных видов знаний имеется в базе знаний системы, тем она интеллектуальней);
- должны обеспечивать возможность пользователю ставить перед системой существенно большее количество задач (в том числе и свободно конструируемых), чем пользовательские интерфейсы не интеллектуальных систем;
- должны как можно более четко отражать семантику предметной области в рамках которой происходит общение с пользователем.

Семантическая технология компонентного проектирования пользовательских интерфейсов интеллектуальных систем [OSTIS, 2010] позволяет создавать пользовательские интерфейсы любой сложности, благодаря тому, что она отвечает ряду требования предъявляемых к ней. Первое требование заключается в снижении сроков проектирования. В настоящее время при проектировании пользовательских интерфейсов большое количество времени тратится на интеграцию уже разработанных компонентов (библиотек). Кроме того существующие технологии для разработки пользовательских интерфейсов требуют достаточно высокий уровень квалификации разработчика. Поэтому вторым требованием, которое предъявляется к семантической технологии проектирования пользовательских интерфейсов интеллектуальных

систем, является снижение требований предъявляемых к начальной квалификации разработчика. Помимо этого технология должна обеспечивать снижение требований предъявляемых к начальной квалификации конечного пользователя. Это немаловажный фактор, так как пользовательские интерфейсы существующих систем значительно отличаются друг от друга, что заставляет пользователей постоянно приспосабливаться к конкретному интерфейсу. Поэтому обеспечение унификации пользовательских интерфейсов на стадии их проектирования может решить эту проблему.

В состав предлагаемой семантической технологии компонентного проектирования пользовательских интерфейсов интеллектуальных систем входит:

- унифицированная семантическая модель пользовательских интерфейсов интеллектуальных систем;
- библиотека совместимых ip-компонентов пользовательских интерфейсов;
- инструментальные средства компонентного проектирования пользовательских интерфейсов;
- методика компонентного проектирования пользовательских интерфейсов;
- методика обучения проектированию пользовательских интерфейсов;
- интеллектуальная help-система по семантической технологии компонентного проектирования пользовательских интерфейсов интеллектуальных систем.

Поставленные задачи соответствуют разработке выше перечисленных подсистем, которые входят в состав технологии.

## 1. Унифицированная семантическая модель пользовательских интерфейсов

В основу семантической модели пользовательских интерфейсов интеллектуальных систем положены следующие принципы:

- пользовательский интерфейс рассматривается как специализированная интеллектуальная система, целью которой является организация взаимодействия пользователя с системой. Как и любая другая интеллектуальная система, он состоит из базы знаний и машины обработки знаний. Далее систему, для которой делается пользовательский интерфейс, будем называть основной системой или же просто – системой.
- в основе графических интерфейсов лежит SCg-код (Semantic Code graphical – который является одним из возможных способов визуального представления SC-кода) [OSTIS, 2010]. Объекты, отображаемые на экране с помощью SCg-кода, будем называть sc.g-элементами. Основной идеей при визуализации является то, что все изображенные на экране объекты, в том числе и элементы управления, являются элементами семантической сети. Другими словами каждому изображенному на экране объекту соответствует sc-узел в базе знаний.
  - возможность явной визуализации связей между объектами. К примеру, отображение отношения декомпозиции, для пунктов меню; явное отображение отношения дочернее окно*;
- выделение семантики в пользовательских действиях. Другими словами, построение четкой типологии пользовательских действий и явное обращение пользователя к их семантике, при работе с системой.

Семантическая модель позволяет логически разделить (декомпозировать) пользовательский интерфейс на подсистемы (компоненты), которые могут быть разработаны независимо друг от друга и использованы в различных системах.

В базе знаний системы может находиться большое количество знаний представленной в различном виде, в том числе информация, записанная в содержимом sc-ссылок [OSTIS, 2010]. Пользовательский интерфейс должен предоставлять возможность ее просмотра и редактирования, для этого вводится два вида подсистем: просмотрщики содержимого sc-ссылок, редакторы содержимого sc-ссылок.

Просмотрщики содержимого sc-ссылок позволяют отображать информацию, записанную в ее содержимом, на некотором внешнем языке. Здесь можно провести аналогию с приложением, которое позволяет просматривать файлы (к примеру, приложение для просмотра видео файлов

или изображений). Только в нашем случае в роли файла выступает содержимое sc-ссылки. Пользователь может сделать запрос на отображение лишь части информации, но никак не более той, что имеется в ее содержимом. Кроме того пользователь может менять способы отображения информации. Например, в случае с визуальным отображением графов, пользователь может менять стилистику размещения.

Редакторы содержимого sc-ссылок позволяют осуществлять редактирование информации представленной в содержимом. Каждый такой редактор состоит из просмотрщика sc-ссылок и набора команд редактирования.

В интеллектуальной системе отображения и редактирования информации записанной в различной форме недостаточно, так как система не может осуществлять ее интеллектуальную обработку. Чтобы решить эту проблему вводится еще два вида интерфейсных подсистем: транслятор содержимого sc-ссылки в SC-код [Голенков и др, 2001] и обратно. Наличие такого рода подсистем позволяет пользовательскому интерфейсу переводить на понятный системе язык (SC-код) информацию, записанную в содержимом sc-ссылок, и осуществлять ее интеллектуальную обработку. Также появляется возможность представить в некоторой внешней форме (к примеру, график функции в виде изображения) знания, имеющиеся в базе знаний или полученные в ходе решения какой-либо задачи.

Описанные выше четыре вида интерфейсных подсистем вполне могут обеспечить пользователю возможность задавать системе вопросы и получать на них ответы в удобном для него виде. Однако существенно более эффективным является редактирование на семантическом уровне. Это означает, что вместо редактирования информации на каком-либо внешнем языке, пользователь редактирует ее представление в базе знаний. К примеру, при построении отрезка ему не обязательно рисовать его на чертеже с помощью специальной команды. Куда более эффективным является команда построения отрезка в базе знаний системы и последующее его автоматическое отображение на любом внешнем языке (в частности геометрический чертеж).

Чтобы обеспечить такую возможность, необходим редактор баз знаний и транслятор базы знаний во внешнее представление. Редактор базы знаний не является частью пользовательского интерфейса, поэтому он не будет рассмотрен. Под транслятором базы знаний во внешнее представление будем понимать интерфейсную подсистему, которая производит отображение пользователю информационных конструкций из базы знаний с помощью внешнего представления. В качестве внешнего представления может выступать речь, графические изображения и т. д. Такой способ организации диалога значительно уменьшает количество используемых компонентов, однако повышает их сложность, за счет необходимости решать задачи связанные с наглядной визуализацией информации (система сама должна решать, как отображать объекты).

Таким образом, в пользовательском интерфейсе выделено 5 типов интерфейсных подсистем, которые решают различные задачи и позволяют организовать диалог пользователя с интеллектуальной системой:

- просмотрщик содержимого sc-ссылок;
- редактор содержимого sc-ссылок;
- транслятор содержимого sc-ссылки в SC-код;
- транслятор из SC-кода в содержимое sc-ссылки;
- транслятор базы знаний во внешнее представление.

Каждую такую интерфейсную подсистему (относящуюся к одному из выше перечисленных типов) будем называть ip-компонентом пользовательского интерфейса. Типология ip-компонентов полностью соответствует выше представленной типологии интерфейсных подсистем. Принципы интеграции ip-компонентов будут описаны далее в библиотеке ip-компонентов.

Графический интерфейс интеллектуальных систем построенных с использованием описываемой технологии представляет собой мультимодальный оконный интерфейс, где под окном понимается скроллируемая область экрана, ограниченная прямоугольником

произвольного размера. Основным графическим языком является SCg-код. Окна отображаются с помощью SCg-кода в виде sc.g-рамок (рисунок 1)

Рисунок 1 - Пример отображения sc.g-рамки

Окна могут находиться в открытом и закрытом состояниях. Открытые окна могут быть пустыми (содержимое которых находится в начальной стадии формирования). Закрытые окна отображают sc-ссылки, которые имеют не пустое содержимое, которое в данный момент не отображено на экране (рисунок 2а).

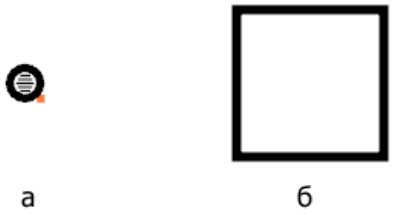


Рисунок 2 - Пример отображения окон. а – отображение закрытого окна,
б – отображение открытого окна

Окно, которое содержит в себе sc.g-конструкцию относится к классу sc.g-окон. По такому же принципу строятся названия и других классов окон. К примеру: класс текстовых окон – окна, в которых содержится текстовая информация; видео-окна – окна, где содержимым является видео и т.д. Окна будем называть в соответствии с названием класса окон, к которому они принадлежат. К примеру, под sc.g-окном будем понимать окно, которое принадлежит классу sc.g-окон.

Главным называется окно системы, в рамках которого осуществляется взаимодействие пользователя с системой, оно принадлежит к классу sc.g-окон. Кроме того оно является синглетоном (не может быть двух главных окон). Главное окно предоставляет возможность общения пользователя с системой, используя SCg-интерфейс. SCg-интерфейс – это целостная часть пользовательского интерфейса, обеспечивающая взаимодействие пользователя с интеллектуальной системой, с помощью SCg-кода. По такому же принципу называются и другие интерфейсы. К примеру: естественно языковой интерфейс (взаимодействие с помощью естественного языка).

При работе с системой пользователь может создавать окна, относящиеся к различным классам (sc.g-окна, видео-окна и т.д.). Такие окна будем называть дочерними окнами, отличительной их особенность является то, что они являются частью sc.g-конструкции в рамках некоторого sc.g-окна. Главное окно не входит во множество дочерних окон. Таким образом, все окна образуют иерархию, которая, в рамках базы знаний пользовательского интерфейса, задается с помощью отношения *дочернее окно** (рисунок 3).


Рисунок 3 - Пример отношения *дочернее окно**

Особенностью SCg-кода является то, что с его помощью можно отображать различного рода элементы управления, которые также являются частью базы знаний (семантической сети) и соответственно представляют собой знания, описанные с помощью SC-кода. Эти знания используются пользовательским интерфейсом для решения задач организации диалога пользователя с системой.

В рамках главного окна, которое относится к классу sc.g-окон, отображаются элементы управления пользовательского интерфейса. SC.g-узлы управления пользовательским интерфейсом – sc-узлы, которые отображаются с помощью SCg-кода на экране и имеет свою, отличную от других, отображаемых на экране sc.g-узлов, операционную семантику. Нажатие кнопки мыши на этих узлах приводит к инициированию некоторых действий. Такие sc.g-узлы могут быть резидентными и нерезидентными. Резидентные sc.g-узлы управления всегда присутствуют (изображаются) на экране в рамках главного окна, в то время, как нерезидентные узлы не всегда присутствуют на экране. При этом на экране могут присутствовать несколько синонимичных sc.g-узлов, которые изображают один и тот же sc-узел, но выглядят по-разному.

Выделяются следующие типы элементов управления:

- элементы управления, изображающие только объект действий. Они используются для отображения объектов находящихся в базе знаний, над которыми можно производить какие-либо действия. Примерами таких элементов управления служат пункты меню, отображающие разделы документации, открытые окна и так далее. По сути, пользователь не может инициировать с помощью них какую-либо команду. Он может лишь их использовать как аргументы для какой-либо команды;
- элементы управления, отображающие классы действий. По сути, представляют собой команды, для которых объект действия не определен. Нажатие клавиши мыши на эти элементы управления приводит к инициированию действия принадлежащего данному классу. К примеру, нажатие клавиши мыши на элемент управления отображающий класс команд “закрыть окно” приводит к инициированию команды, которая принадлежит данному классу и осуществляет закрытие указанного пользователем окна. Элементы управления, относящиеся к данному типу, отображаются на экране в виде знаков атомарных классов пользовательских интерфейсных команд. Они могут отображаться в составе меню и отдельно (являются наиболее часто используемыми командами). При отображении в составе меню, они появляются на экране при инициировании команды открытия пункта меню и таким образом являются не резидентными sc.g-узлами. В свою очередь класс атомарных команд делится на следующие подклассы:
  - команда-вопрос – команд, которая инициирует вопрос системе;
  - команда редактирования – команда, которая инициирует операцию редактирования информационной конструкции (создание или удаление объектов, изменение типа объектов и т. д.);
  - команда просмотра – команда, которая инициирует действия связанные с информационных конструкций (навигация, масштабирование и т. д.).

- элементы управления, отображающие конкретные действия. Они отображают конкретные команды, для которых уже определен объект действия. Примером такого элемента управления является команда закрытия конкретного окна (в современных интерфейсах отображаемая в правом верхнем углу, в виде крестика).

Структура главного окна представлена на рисунке 4. В рамках этого окна имеется главное меню (на рисунке 4 изображено верхней части окна), которое состоит из отображения атомарных и неатомарных пользовательских команд (команда пользовательского интерфейса, при инициировании которой выводится ее *декомпозиция**).

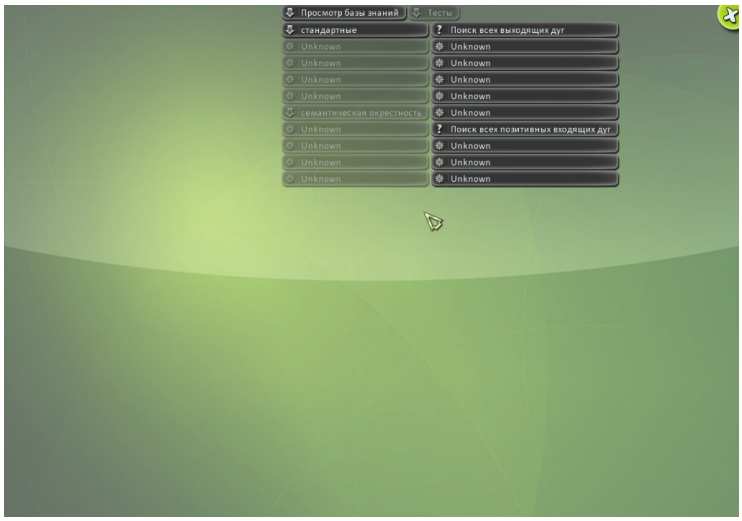


Рисунок 4 – Пример главного окна графического интерфейса.

Каждому отображаемому на экране элементу управления ставится в соответствие графический примитив, который соответствует его классу. К примеру, атомарные и неатомарные команды отображаются в виде прямоугольника с закругленными краями (рисунок 5). При этом текстовый идентификатор объекта отображается внутри прямоугольника.


Рисунок 5 – Пример отображения пользовательской команды

В левой части прямоугольника отображается изображение, которое явно указывает на принадлежность команды к определенному классу команд. На рисунке 6 показан пример отображения неатомарной команды.


Рисунок 6 – Пример отображения неатомарной пользовательской команды

При инициировании пользователем команды, в базе знаний генерируется конструкция, которая является записью данной команды с помощью SC-кода. После этого в машине обработки знаний пользовательского интерфейса инициируется операция, отвечающая за выполнение указанной команды. Пример описания команды-вопроса представлен на рисунке 7.

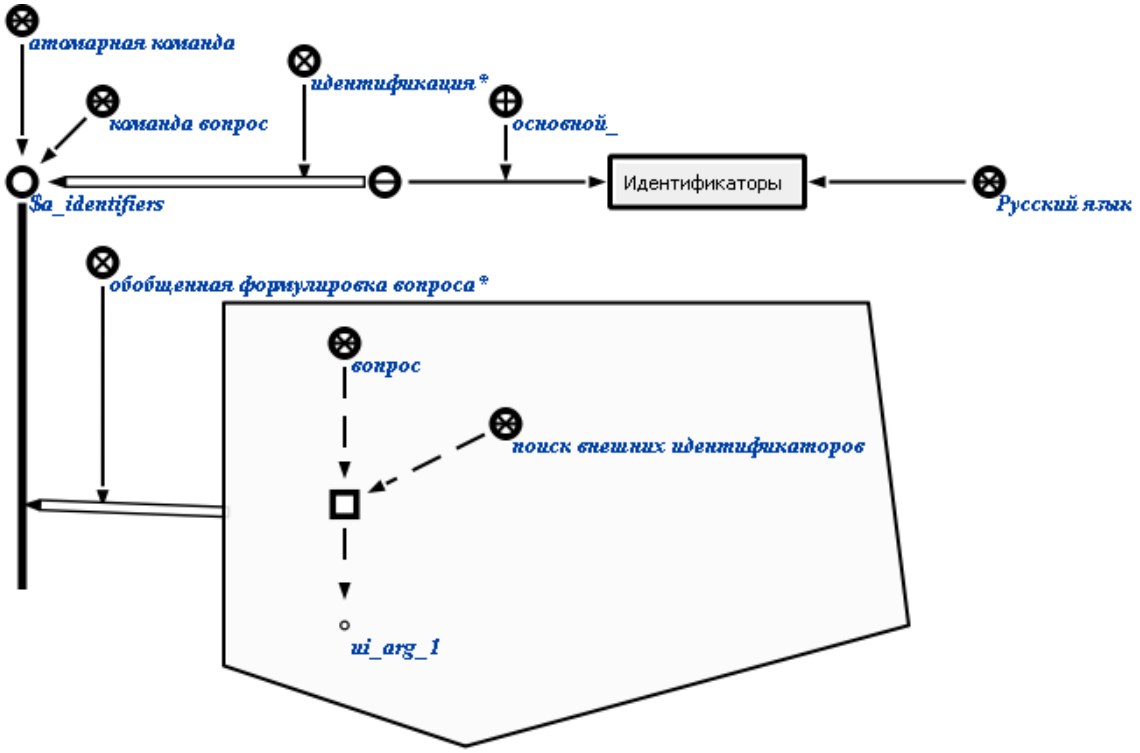


Рисунок 7 – Пример описания команды-вопроса

Использование команд пользовательского интерфейса сводится к следующему алгоритму: пользователь указывает аргументы команды, инициирует команду (нажимая левую клавишу мыши на sc.g-узле, обозначающем команду). Это позволяет унифицировать работу с пользовательским интерфейсом. Однако зачастую такая форма организации может быть неудобной для пользователя, поэтому у него также есть возможность использовать не только за ранее заготовленные элементы управления, но и формулировать вопросы на любом внешнем языке, доступном в системе (к примеру, команды могут быть сформулированы с помощью речи или на естественном языке). Важным является и то, что это дает возможность системе проанализировать действия пользователя и тем самым адаптировать под него интерфейс. Для этого все инициируемые пользователем действия и команды записываются в протокол пользовательских действий. Данный протокол кроме всего прочего позволяет организовать процедуру отмены пользовательских действий.

Такой способ организации интерфейса, когда все отображаемые пользователю объекты, являются частью одной семантической сети, дает ему возможность задавать вопросы, которые касаются не только предметной области, но и пользовательского интерфейса. Что в свою очередь позволяет добавить в проектируемую систему достаточно мощную справочную информацию по работе с ней. Кроме того эта же особенность пользовательского интерфейса позволяет системе управлять самой собой. И данная возможность опять же позволяет значительно повысить эффективность работы с системой, так как у пользователя появляется возможность попросить систему продемонстрировать пример работы с той или иной командой.

### 1.1. База знаний пользовательского интерфейса

База знаний пользовательского интерфейса включает в себя следующую информацию:

- описание команд пользовательского интерфейса – описываются классы команд пользовательского интерфейса, которые служат для инициирования пользователем каких-либо действий в системе. Пример описания представлен на рисунке 7;
- описание структуры главного меню;
- описание внешних языков представления знаний;

- описание используемых компонентов – используются интерфейсом для организации диалога с пользователем. К примеру, при установке содержимого sc-рамки пользователю отображается список редакторов, которые имеются в базе знаний;
- протокол пользовательских действий – строится во время работы системы. Необходим для построения портрета пользователя, а также обеспечения отмены пользовательских действий;
- портрет пользователя – строится на основании анализа действий пользователя. Включает в себя всю необходимую информацию о пользователе. В качестве такой информации может выступать предпочитаемая стилистика размещения, наиболее часто используемые внешние языки и т. д.;

### 1.2. Машина обработки знаний пользовательского интерфейса

Машина обработки знаний пользовательского интерфейса имеет свою специфику – кроме обычных операций обработки знаний в ней присутствуют “эффекторные” и “рецепторные” операции. “Эффекторные” операции осуществляют вывод информации пользователю, реагируя на событие в памяти системы. “Рецепторные” операции реагируют на события пользователя (нажатие клавиши, перемещение мыши) и изменяют состояние памяти системы.

В машину обработки знаний пользовательского интерфейса обязательно входят следующие операции:

- Операции работы с командами:
  - добавление дочерней команды в неатомарную команду;
  - удаление дочерней команды из неатомарной команды;
- Операции работы с компонентами:
  - добавление компонента;
  - удаление компонента;
- Операции работы с окнами:
  - создание окна указанного типа;
  - удаление окна;
  - открытие/закрытие окна;
- “Эффекторные” операции:
  - инициирование вывода ответа на вопрос заданный пользователем;
  - вывод информационной конструкции пользователю;
- “Рецепторные” операции (список может быть расширен в зависимости от используемых внешних устройств):
  - нажатие кнопки на клавиатуре;
  - нажатие клавиши мыши;
  - вращение колеса прокрутки мыши;

## 2. Библиотека совместимых ip-компонентов пользовательских интерфейсов интеллектуальных систем

Большое разнообразие пользовательских интерфейсов влечет за собой разработку большого числа ip-компонентов (под ip-компонентом понимается часть пользовательского интерфейса, которая может быть интегрирована и использована в различных системах). Стоит отметить, что ip-компоненты могут состоять из других ip-компонентов (уровень вложенности не ограничен). Таким образом, в качестве ip-компонентов, могут выступать уже спроектированные пользовательские интерфейсы. Большое число ip-компонентов создает проблему их хранения и поиска. Чтобы решить эту проблему, в технологию включена библиотека совместимых ip-компонентов пользовательских интерфейсов (далее библиотека ip-компонентов). Она

представляет собой специализированную интеллектуальную систему, которая решает следующие задачи: хранение, поиск, добавление, удаление (поддержка актуальности) и сравнение ip-компонентов пользовательских интерфейсов.

В рамках библиотеки, все ip-компоненты специфицируются и классифицируются. Их классификация производится по различным признакам. К примеру, в зависимости от решаемых задач ip-компоненты делятся на следующие классы:

- просмотрщик содержимого sc-ссылок;
- редактор содержимого sc-ссылок;
- транслятор содержимого sc-ссылки в SC-код;
- транслятор из SC-кода в содержимое sc-ссылки;
- транслятор базы знаний во внешнее представление.

Технология позволяет интегрировать в качестве ip-компонентов редакторы и просмотрщики, разработанные с использованием других технологий (далее их будем называть инородными ip-компонентами пользовательского интерфейса). В основном они используются для просмотра и редактирования содержимого sc-ссылок. Это значительно позволяет сэкономить время на их разработке.

База знаний библиотеки ip-компонентов включает в себя следующую информацию:

- спецификация ip-компонентов;
- история версий ip-компонентов;
- статистика использования ip-компонентов.

Машина обработки знаний состоит из следующего набора операций:

- поиск ip-компонента, по его спецификации;
- добавление нового ip-компонента;
- удаление ip-компонента;
- сравнение двух ip-компонентов.

Пользовательский интерфейс библиотеки ip-компонентов строится на основе SCg-интерфейса (комплекс информационно-программных средств обеспечивающих общение интеллектуальных систем с пользователями на основе SCg-кода, как способа внешнего представления информации). Однако, это не исключает возможность использования других способов диалога пользователя с библиотекой ip-компонентов.

В рамках библиотеки ip-компонентов могут содержаться различные версии и модификации какого-либо ip-компонента. К примеру, ip-компонент просмотра sc.g-конструкций может иметь модификации, для отображения в которых может использоваться двумерная, трехмерная или же многослойная визуализация, при этом каждая из модификаций компонента может иметь различные версии.

Использование библиотеки ip-компонентов при проектировании пользовательского интерфейса прикладной системы позволяет значительно сократить сроки проектирования, а также снизить требования, предъявляемые к начальной квалификации разработчика. Это достигается за счет проектирования пользовательского интерфейса из уже заранее заготовленных модулей, что также позволяет повысить качество проектируемого интерфейса.

При таком подходе основной проблемой является проблема интеграции компонентов между собой. Для ее решения предлагается осуществлять взаимодействие между ip-компонентами через базу знаний. Таким образом, взаимодействуя между собой, компоненты могут лишь использовать общие ключевые узлы (понятия) в базе знаний. Такой способ интеграции компонентов позволяет разрабатывать их параллельно и независимо друг от друга, что значительно сокращает сроки проектирования.

Одной из сложностей при проектировании интерфейса является его платформенная ориентация. Чтобы решить данную проблему в библиотеку ip-компонентов входят

компоненты, которые реализуют ядро пользовательских интерфейсов (собственно реализация принципов заложенных в семантической модели). Ядро пользовательских интерфейсов включает в себя весь набор необходимых интерфейсных операций, а также осуществляет интеграцию используемых ip-компонентов между собой. Таким образом, наличие ядра пользовательских интерфейсов в библиотеке ip-компонентов, позволяет разработчику не задумываться о погрузке разработанного им интерфейса на конкретную платформу. Стоит отметить, что разработанные как sc-системы ip-компоненты являются максимально независимыми от платформы, в отличие от инородных ip-компонентов.

## 3. Инструментальные средства компонентного проектирования пользовательских интерфейсов интеллектуальных систем

Интегрированные средства проектирования пользовательских интерфейсов интеллектуальных систем решают задачу автоматизации и интеллектуализации процесса проектирования. В их состав входят инструментальные средства визуального проектирования, средства отладки и верификации проектируемых интерфейсов, библиотека ip-компонентов пользовательских интерфейсов и help-система. Интеграция указанных систем в рамках интегрированной среды, позволяет значительно ускорить процесс проектирования, а также повысить качество проектируемых интерфейсов.

В основе инструментальных средств визуального проектирования лежит пользовательский SCg-интерфейс. Это дает разработчику возможность проектирования пользовательских интерфейсов с использованием известной технологии WYSIWYG (What You See Is What You Get – «что вы видите то и получите», то есть результат проектирования выглядит так же, как и прототип во время разработки).

Инструментальные средства верификации и отладки, пользовательских интерфейсов, представляют собой специализированную интеллектуальную систему, которая позволяет находить и устранять ошибки в проектируемом пользовательском интерфейсе. Эта система направлена на автоматизацию и последующую интеллектуализацию процесса верификации и отладки проектируемого интерфейса. Для нее выделено три уровня интеллектуализации:

- на первом уровне средства верификации и отладки позволяют осуществлять поиск ошибок в проектируемом интерфейсе;
- на втором уровне интеллектуализации добавляется возможность анализа и оптимизации проектируемого интерфейса;
- на третьем уровне интеллектуализации появляется возможность автоматического исправления ошибок.

## 4. Методика компонентного проектирования пользовательских интерфейсов

Методика проектирования пользовательских интерфейсов является важной частью технологии, так как она описывает сам процесс проектирования.

На первом этапе разработчику необходимо специфицировать проектируемый пользовательский интерфейс. Спецификация включает в себя список задач решаемых интерфейсом, описание внешних языков представления знаний.

На этапе задачно-ориентированной декомпозиции специфицированный разработчиком интерфейс разбивается на интерфейсные подсистемы, которые могут разрабатываться параллельно. Это позволяет сократить сроки проектирования пользовательского интерфейса. Целесообразно проводить разбиение таким образом, чтобы максимальное количество подсистем уже имелось в библиотеке ip-компонентов пользовательских интерфейсов.

После уточнения функциональных возможностей разрабатываемых подсистем производится их разработка с использованием семантической технологии проектирования интеллектуальных систем.

После разработки пользовательского интерфейса разработчик выделяет из него типовые фрагменты и специфицируя их необходимым образом включает в библиотеку ip-компонентов, тем самым внося вклад в развитие технологии.

## 5. Интеллектуальная help-система по технологии

Интеллектуальная help-система, по семантической технологии проектирования пользовательских интерфейсов интеллектуальных систем, решает задачу помощи разработчикам в процессе проектирования пользовательских интерфейсов. Данная интеллектуальная система строится с использованием семантической технологии проектирования интеллектуальных систем. Её база знаний содержит следующие разделы:

- унифицированная семантическая модель пользовательских интерфейсов интеллектуальных систем. В данном разделе содержится полное формальное описание модели пользовательских интерфейсов, это дает возможность пользователю (разработчику) значительно быстрее усвоить теорию и понять принципы лежащие в основе проектируемых им интерфейсов;
- библиотека совместимых ip-компонентов пользовательских интерфейсов интеллектуальных систем. В этом разделе приводится полное формальное описание библиотеки ip-компонентов. Это позволяет разработчику значительно быстрее освоить принципы работы с библиотекой и тем самым сократить сроки проектирования;
- интегрированные средства проектирования пользовательских интерфейсов. Раздел базы знаний посвященный описанию средств, которые позволяют автоматизировать и интеллектуализировать процесс проектирования;
- методика проектирования пользовательских интерфейсов интеллектуальных систем. Содержит детальное описание процесса проектирования, что в свою очередь позволяет пользователю получить ответ на такой вопрос: “Что и как делать дальше?”;
- методика обучения проектированию пользовательских интерфейсов интеллектуальных систем. Данный раздел базы знаний полностью посвящен процессу обучения проектированию. Он содержит информацию, которая должна помочь пользователю в освоении самой технологии;
- help-система по семантической технологии проектирования пользовательских интерфейсов интеллектуальных систем. Раздел, который призван помочь пользователю освоить саму help-систему.

Включение интеллектуальной help-системы в состав технологии позволяет не только сократить сроки проектирования, но и снизить требования, предъявляемые к начальной квалификации разработчика.

## Заключение

В настоящее время ведется работа по реализации описанной выше технологии. Однако на данном этапе уже можно сказать, что она позволяет быстро и качественно проектировать пользовательские интерфейсы интеллектуальных систем. Основными ее достоинством являются простота и скорость проектирования. Все это достигается за счет использования компонентного подхода при проектировании и использования унифицированной модели пользовательского интерфейса.

Работа поддержана грантом БРФФИ Ф10М-085.

## Библиографический список

**[OSTIS, 2010]** Открытая семантическая технология проектирования интеллектуальных систем [Электронный ресурс]. – 2010. - Режим доступа: http://ostis.net. – Дата доступа: 01.12.2010

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

**[Голенков и др, 2009]** Проектирование пользовательских интерфейсов интеллектуальных систем / В. В. Голенко, В. А. Житко, Д. Г. Колб, Д. Н. Корончик// Нечеткие системы и мягкие вычисления (НСМВ - 2009). Сборник статей Третьей Всероссийской научной конференции. Том II. - с. 181-188.

**[Грибова и др., 2005]** Использование методов искусственного интеллекта для проектирования пользовательского интерфейса / В.В. Грибова, А.С. Клещев // Информационные технологии. №8. — 2005. — с.58-62.

**[Грибова и др., 2008]** Управление автоматизацией проектирования пользовательских интерфейсов на основе знаний / В. В. Грибова, Р. С. Кисленок // Труды конференции КИИ-2008. Том III. –с. 266 – 274.

**[Касьянов и др., 2003]** Графы в программировании: обработка, визуализация и применение / В. Н. Касьянов, В. А. Евстигнеев; - Научное издание, 2003.

**[Курзанцева, 2008]** Об адаптивном интеллектуальном интерфейсе “Пользователь – система массового применения” / Л.И. Курзанцева // Комп’ютерні засоби, мережі та системи №7, 2008. – с. 110 – 116.

**[Поспелов, 1989]** Интеллектуальные интерфейсы для ЭВМ новых поколений / Поспелов Д.А. // Электронная вычислительная техника. - М.: Радио и связь, 1989. - Вып. 3. - с. 4-20.

**[Fei, 2001]** Multi-Modal Human Interactions with an Intelligent Interface Utilizing Images, Sounds and Force Feedback / Fei He, Arvin Agah // Journal of Intelligent and Robotic Systems. – Kluwer Academic Publisher, 2001. – p. 171 – 190.

**[Myers B.A., 1992]** Survey on User Interface Programming, Proceedings SIGCHI'92: Human Factors in Computing Systems / Myers B.A., Rosson M.B. - Monterrey, CA, 1992. - P. 195-202.

**[Patrick, 2003]** Intelligent user interfaces: introduction and survey / Patrick A.M. – Delft University of Technology, 2003.