OSTIS-2012
(Open Semantic Technologies for Intelligent Systems)

УДК 004.822:514

СЕМАНТИЧЕСКИЕ МОДЕЛИ МУЛЬТИМОДАЛЬНЫХ ПОЛЬЗОВАТЕЛЬСКИХ ИНТЕРФЕЙСОВ И СЕМАНТИЧЕСКАЯ ТЕХНОЛОГИЯ ИХ ПРОЕКТИРОВАНИЯ

Корончик Д. Н.

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

denis.koronchik@gmail.com

В работе описана технология проектирования пользовательских интерфейсов интеллектуальных систем. Приводится описание семантической модели пользовательских интерфейсов, которая является основной описываемой технологии. Предложенная в данной работе технология позволяет проектировать мультимодальные пользовательские интерфейсы на основе готовых совместимых компонентов.
Ключевые слова: интеллектуальный пользовательский интерфейс, мультимодальный пользовательский интерфейс, семантическая модель, семантическая технология.

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.

Проектируемые с помощью современных технологий, пользовательские интерфейсы представляют собой достаточно сложные системы. Основной проблемой в них является то, что пользователю, имеющему низкий уровень квалификации, сложно работать с ними. Это в свою очередь уменьшает количество пользователей и снижает эффективность их эксплуатации. Одним из важных факторов в этом является то, что все такие пользовательские интерфейсы имеют различные принципы организации. Кроме того для их освоения необходимо читать большое количество документации.

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;
- должны как можно более четко отражать семантику предметной области в рамках которой происходит общение с пользователем.

Пользовательский интерфейс является единственным способом взаимодействия пользователя с программной системой. Поэтому они должны быть достаточно простыми и легкими в освоении [Поспелов, 1989]. Предлагаемая в данной работе технология направлена на решение описанных выше проблем.

1. Общие положения семантической технологии проектирования пользовательских интерфейсов

В основе предлагаемой семантической технологии проектирования пользовательских интерфейсов, лежат следующие принципы:

- разработка унифицированной модели пользовательских интерфейсов согласованной с общей теорией интеллектуальных систем (что не только сокращает сроки разработки, но и упрощает интеграцию пользовательского интерфейса в состав системы). В основе предлагаемой модели пользовательских интерфейсов лежит семантическая модель, основу которой составляет унифицированное представление информации с помощью SC-кода, описанного в [Голенков и др, 2001];

- повторное (многократное) использование уже разработанных фрагментов пользовательских интерфейсов (что обеспечивает не только сокращение сроков разработки благодаря многократному повторному использованию типовых компонентов пользовательских интерфейсов, но и снижает требования к квалификации разработчиков, поскольку это переводит проектирование на более высокий уровень – на уровень модульного сборочного проектирования). Фрагменты пользовательского интерфейса, которые являются достаточно унифицированными и могут быть использованы многократно в различных системах, называются компонентами пользовательского интерфейса. Для хранения, компонентов используется библиотека компонентов пользовательских интерфейсов;
- повышение уровня автоматизации и последующей интеллектуализации процесса проектирования (что сокращает сроки разработки, упрощает интеграцию пользовательского интерфейса в состав интеллектуальной системы). Для проектирования пользовательских интерфейсов используются инструментальные средства визуального проектирования, а также средства верификации и отладки, которые также являются интеллектуальными системами;
- создание интеллектуальной help-системы обеспечивающей консультацию разработчиков пользовательских интерфейсов по всем аспектам технологии их проектирования (что не только снижает сроки проектирования, но и снижает требования, предъявляемые к начальной квалификации разработчиков). Интеллектуальная help-система является sc-системой, основной задачей которой является формирование у разработчика навыков по проектированию пользовательских интерфейсов.

Таким образом, технология проектирования пользовательских интерфейсов включает в себя:
- семантическую модель пользовательских интерфейсов;
- библиотеку совместимых компонентов;
- инструментальные средства проектирования пользовательских интерфейсов:
  - интеллектуальные средства визуального проектирования;
  - интеллектуальные средства отладки и верификации;
- методику проектирования пользовательских интерфейсов;
- методику обучения проектированию пользовательских интерфейсов;
- help-систему по проектированию пользовательских интерфейсов.

2. Семантическая модель мультимодального пользовательского интерфейса

Основной частью технологии проектирования пользовательских интерфейсов является семантическая модель. Она описывает принципы, которые лежат в основе проектируемых интерфейсов, что позволяет унифицировать их. В основу семантической модели положены следующие принципы:

- пользовательский интерфейс рассматривается как специализированная интеллектуальная система, которая направлена на получение сообщений от пользователя и вывода ему ответов системы. Другими словами, основной задачей пользовательского интерфейса является перевод сообщения от пользователя, полученного на некотором внешнем языке, на язык понятный системе, а также вывод ответа системы на некотором внешнем языке, понятном пользователю;
- 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);
- выделение семантики в пользовательских действиях, с последующим анализом, а также их унификация и четкая типология.

Так как пользовательский интерфейс представляет собой интеллектуальную систему, построенную с помощью семантической технологии компонентного проектирования интеллектуальных систем, то он точно так же как и любая другая система строится с использованием компонентов.

Выделяются следующие типы компонентов:
- трансляторы с текстов различных внешних языков в тексты SC-кода;
- трансляторы с текстов SC-кода в тексты различных внешних языков;
- компоненты вывода информационных конструкций пользователю;
- компоненты ввода информационных конструкций;

Каждый компонент пользовательского интерфейса состоит из некоторой базы знаний необходимой для его работы и набора sc-операций.

Графический интерфейс интеллектуальных систем представляет собой мультимодальный оконный интерфейс. Под окном будем понимать скроллируемую область экрана, которая ограничена

прямоугольником произвольного размера.

Взаимодействие пользователя с системой осуществляется в рамках главного окна. Главное окно принадлежит к множеству *sc.g-окно* (окна, внутри которых для визуализации знаний используется SCg-код). В рамках главного окна могут присутствовать другие окна. Их будем называть дочерними окнами. Визуализируются они с помощью sc.g-рамок (рисунок 1) внутри которых отображается содержимое окна на одном из внешних языков.

Рисунок 1 – Пример изображения sc.g-рамки

С помощью sc.g-рамок отображаются окна в развернутом виде. В свернутом виде они изображаются с помощью sc.g-узла с квадратом в правом нижнем углу 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. Они являются частью sc.g-конструкции в рамках некоторого *sc.g-окна* (в частности, главного окна). Главное окно не может быть дочерним окном по отношению к другим окнам. И одно окно не может иметь несколько родительских окон. Таким образом, они образуют древовидную иерархию, которая, в рамках базы знаний пользовательского интерфейса, задается с помощью отношения *дочернее окно** (рисунок 2).

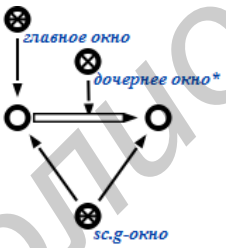

Рисунок 2 – Пример использования отношения *дочернее окно**

Графический пользовательский интерфейс строится с использованием уже разработанных компонентов. К компонентам вывода информационных конструкций относятся просмотщики, которые в свою очередь могут быть двух типов:

- просмотщики содержимого sc-ссылок. Позволяют просматривать содержимое sc-ссылок (файлов) записанное в некотором формате (не SC-код);
- просмотрщики фрагментов базы знаний представленных с помощью SC-кода. Позволяют просматривать с помощью внешнего языка информационные конструкции, представленные в SC-коде (к примеру, фрагмент базы знаний, описывающий геометрический чертеж с помощью SC-кода может изображаться в виде sc.g-текста и в виде геометрического чертежа).

К компонентам ввода информационных конструкций относятся редакторы. Они позволяют редактировать содержимое некоторой sc-ссылки. Главное окно представляет собой просмотрщик фрагмента базы знаний с помощью SCg-кода и набора команд редактирования базы знаний. Другими словами при инициировании команд редактирования в рамках главного окна происходит редактирование не на уровне внешнего представления, а напрямую в базе знаний, на уровне SC-кода, а эти изменения отображаются с помощью просмотрщика.

При таком подходе основной проблемой является проблема интеграции компонентов между собой. Для ее решения предлагается осуществлять взаимодействие между 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 с минимальными зависимостями, что значительно сокращает сроки проектирования.

Диалог пользователя с системой осуществляется с использованием SCg-кода. Пользователь формирует сообщения системе, явно рисуя их с помощью SCg-кода (рисунок 3) и инициируя команду погрузки в базу знаний. Такой способ формирования сообщений назовем базовым. Ответы, как и вопросы, выводятся с помощью SCg-кода (в виде sc.g-конструкций). Частным видом ответа является sc.g-конструкция, состоящая из одной sc.g-рамки, содержимое которой представляется на каком-то внешнем языке. Другими словами диалог сводиться к обмену сообщениями (представленными с помощью SCg-кода) между пользователем и системой.

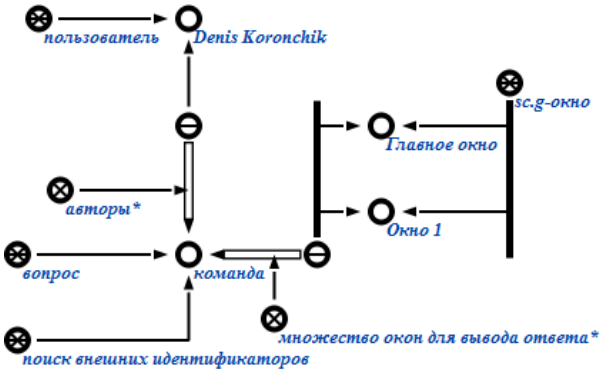

Рисунок 3 – Пример сформированного вопроса с помощью SCg-кода.

На рисунке 3, задается вопрос: *Какие внешние идентификаторы имеются у узла с основным идентификатором "команда"*. Помимо самой сути вопроса, к нему еще добавляется информация об

авторе вопроса (в нашем случае это пользователь, поэтому ответ необходимо будет выдать ему). Кроме этого дополнительно указывается множество окон, в которые необходимо вывести ответ (в качестве таких окон могут выступать окна не принадлежащие множеству sc.g-окон, что дает возможность получать ответ на любом внешнем языке). Таким же образом формируются различного рода команды (сворачивание, разворачивание, создание, удаление окна и т. д.). Такой способ формирования команд и вопросов является базовым и универсальным.

В процессе формирования сообщений таким способом, пользователю приходиться выполнять часто повторяющиеся элементарные пользовательские действия (самые простейшие манипуляции с устройствами ввода: нажатие клавиши, перемещение мыши и т.д.). Очевидно, что постоянный набор сообщений вручную с помощью SCg-кода отнимает много времени и не является очень эффективным, по причине выполнения большого количества повторяющихся элементарных пользовательских действий. Можно ускорить процесс набора сообщений, за счет улучшения и расширения команд редактирования sc.g-конструкций. Но это не даст значительного прироста, потому что при формировании сообщения, необходимо будет указывать дополнительную информацию (к примеру, для каждого вопроса надо указывать авторов и множество окон для вывода).

Введем в модель такое понятие как пользовательская команда. Пользовательская команда - последовательность элементарных пользовательских действий, инициирующая некоторое действие системы. К элементарным действиям пользователя относятся: перемещение мыши, нажатие клавиш мыши и клавиатуры. Выделяются следующие классы пользовательских команд по типу инициируемого действия:

- команды, которые инициируют вопрос в системе;
- команды, которые инициируют действия связанные с редактированием sc-текстов;
- команды, которые инициируют действия связанные с просмотром sc-текстов;
- команды, которые инициируют вывод *декомпозиции** или *разбиения** объекта.

По способу определения аргументов, выделяются следующие классы пользовательских команд:

- команды с заранее определенными аргументами (объектами, над которыми будет производиться инициируемое действие);
- команды с дополнительно указываемыми аргументами.

Как и вся другая информация, пользовательские команды представлены с помощью SC-кода в базе знаний. При визуализации с помощью SCg-кода, они представляются в виде элементов управления. Рекомендуется изображать элементы управления в виде прямоугольников, внутри которых присутствует графический или текстовый идентификатор команды. Чтобы различать различные классы команд кроме идентификатора команды в прямоугольнике изображается графический идентификатор класса команды (пример на рисунке 4)

Рисунок 4 – Пример изображения команды, которая инициируют вывод *декомпозиции** или *разбиения** объекта.

Инициирование пользовательской команды происходит при нажатии левой клавиши мыши на sc.g-узле, который её изображает. Все инициируемые пользователем команды попадают в базу знаний, где хранится их четкая последовательность. Это дает возможность их последующей обработки и анализа. Для анализа действий пользователя выделяется специализированная система – система управления диалогом. Именно в этой системе принимаются решения о том, каким образом строить диалог между пользователем и системой.

В современных пользовательских интерфейсах существующих приложений, как и в предлагаемом подходе, также есть команды, для инициирования которых необходимо указывать аргументы и команды, для которых аргументы уже заранее определены. В качестве примера команд с известными аргументами (объектами действий) можно привести команды закрытия окна (окно заранее известно), команды сворачивания окна и т. д. В качестве команд, для которых необходимо указать аргументы, можно привести следующие: указание фрагмента текста при форматировании в редакторе Word и т. д. Проблемой является то, что в различных системах указание аргументов происходит различным. Это в свою очередь требует от пользователя некоторого времени для освоения. В предлагаемой модели указание аргументов команд происходит явно (при этом пользователь всегда может узнать, какие аргументы необходимы для инициирования команды, задав соответствующий вопрос системе). Таким образом, процесс инициирования команды сводится к следующей последовательности действий:

- указание аргументов команды (в настоящий момент это делается левой клавишей мыши, с зажатой клавишей Alt);
- инициирование команды нажатием левой клавиши мыши (без зажатой клавиши Alt) на sc.g-узле изображающем её.

База знаний пользовательского интерфейса состоит из следующих разделов:

- Описание синтаксиса и семантики всех используемых внешних языков;
- Описание пользовательских команд;
- Описание принципов работы самого пользовательского интерфейса;
- Описание используемых компонентов – используются интерфейсом для организации

диалога с пользователем. К примеру, при установке содержимого sc-рамки пользователю отображается список редакторов, которые позволяют его редактировать;
- Протокол действий пользователя.

Машина обработки знаний интерфейса состоит из набора sc-операций. Кроме sc-операций, работающих только над sc-памятью, она содержит эффекторные и рецепторные операции. Эффекторные sc-операции, реагируя на изменения в sc-памяти, производят изменения во внешней среде (вывод информации). Рецепторные sc-операции, реагируя на изменения во внешней среде, производят изменения в sc-памяти (ввод информации).

Машина обработки знаний пользовательского интерфейса состоит из следующих типов sc-операций:
- Операции работы с окнами:
  - создание окна указанного типа;
  - удаление окна;
  - открытие/закрытие окна;
- Эффекторные sc-операции:
  - инициирование вывода ответа на вопрос заданный пользователем;
  - вывод информационной конструкции пользователю (на экран, в наушники и т. д.);
- Операции эмуляции элементарных действий пользователя:
  - Перемещение мыши;
  - Нажатие клавиш на клавиатуре;
  - Вращение колеса прокрутки.
- Рецепторные sc-операции (список может быть расширен в зависимости от используемых внешних устройств):
  - нажатие кнопки на клавиатуре;
  - нажатие клавиши мыши;
  - вращение колеса прокрутки мыши;
  - sc-операции трансляции информации из SC-кода во внешнее представление и обратно.

Описанные выше принципы построения пользовательского интерфейса позволяют проектировать пользовательские интерфейсы достаточно быстро и легко. При этом созданные таким образом пользовательские интерфейсы с точки зрения пользователя внешне мало чем отличаются от современных пользовательских интерфейсов:
- проектируемый пользовательский интерфейс является оконным, что присутствует во всех системах. Главное окно можно сравнить с рабочим столом операционной системы. В рамках этого окна создаются дочерние окна, в которых происходит просмотр или редактирование некоторых файлов;
- в рамках главного окна (рабочего стола) присутствуют некоторые команды, которые могут быть инициированы пользователем с указанием аргументов аргументами или же без них.

Однако предлагаемый подход имеет ряд преимуществ. Так как в основе предлагаемых графических пользовательских интерфейсов лежит SCg-код, то у пользователя появляется возможность указывать и элементы управления в качестве аргументов команд. Это значительно снижает требования, предъявляемые к начальной квалификации пользователя. Ему лишь необходимо знать, как задать вопрос. Умея задавать вопросы, он может легко изучить команды и научиться работать с пользовательским интерфейсом (в современных системах для этого приходиться читать документацию). Унификация процесса формирования пользовательских команд, также упрощает освоение пользовательского интерфейса конечным пользователем, так как при переходе от пользовательского интерфейса одной системы к интерфейсу другой ему не нужно заново учиться этому. Благодаря тому, что база знаний пользовательского интерфейса содержит описание команд и используется SCg-код, появляется возможность производить демонстрации работы с теми или иными командами. Пользователь может попросить систему показать, как пользоваться той или иной командой. При этом система, основываясь на описании команды, построит последовательность элементарных пользовательских действий, которая потом будет выполнена специализированными sc-операциями. Другими словами пользователю будет явно показано, как это сделать (система возьмет под свой контроль перемещение мыши и клавиатуру). Это также дает целый ряд преимуществ, при тестировании проектируемых интерфейсов. Так как для этого не надо записывать огромные протоколы с явным указанием элементарных действий (как это делается при тестировании современных интерфейсов). Достаточно лишь описать последовательность команд, остальное сделает сама система.

Полное описание пользовательского интерфейса с помощью SC-кода в базе знаний позволяет также решить еще одну очень важную задачу. Сейчас очень остро стоит проблема проектирования документации к разрабатываемым системам и их пользовательскому интерфейсу. По сути происходит следующее:
- проектировщик (дизайнер) описывает пользовательский интерфейс (создает техническое задание), в котором описываются все команды, внешние языки и т. д.;
- разработчик (программист) реализует описанные команды и принципы;
- технический писатель пишет по ним документацию.

На всех трех этапах одни и те же вещи описываются на различных языках. В предлагаемом подходе, описания команд, которые делаются разработчиком на формальном языке. при описании проектируемого интерфейса, сразу же и является частью справочной системы, которая в свою очередь позволяет задавать вопросы и получать ассоциативный доступ к хранимой информации (чего нет в современных справочных системах). Это

значительно сокращает сроки проектирования. Так как уже разработанный интерфейс и является большой частью справочной системы по его эксплуатации.

Разбиение пользовательского интерфейса на компоненты, которые взаимодействуют между собой через общую sc-память (базу знаний), используя общие ключевые узлы и набор sc-операций, сводит зависимость между ними к минимуму. Снижение зависимостей между проектируемыми компонентами значительно сокращает их срок разработки, и упрощает дальнейшую поддержку.

Благодаря тому, что все действия пользователя семантически интерпретируются и записываются в протокол, появляется возможность их анализа. Это делает возможным помощь пользователю при освоении пользовательского интерфейса (к примеру, система может подсказать пользователю, что некоторые действия можно делать гораздо проще). Также это дает возможность системе подстраивать пользовательский интерфейс под конкретного пользователя (показывать наиболее часто используемые команды, использовать более удобные принципы размещения элементов управления и т. д.)

3. Библиотека совместимых компонентов пользовательских интерфейсов

Большое разнообразие пользовательских интерфейсов влечет за собой разработку большого числа компонентов (под компонентом понимается часть пользовательского интерфейса, которая может быть интегрирована и использована в различных системах). Стоит отметить, что компоненты могут состоять из других компонентов (уровень вложенности не ограничен). Таким образом, в качестве компонентов, могут выступать уже спроектированные пользовательские интерфейсы. Большое число компонентов создает проблему их хранения и поиска. Чтобы решить эту проблему, в технологию включена библиотека совместимых компонентов пользовательских интерфейсов (далее библиотека компонентов). Она представляет собой специализированную интеллектуальную систему, которая решает следующие задачи: хранение, поиск, добавление, удаление (поддержка актуальности) и сравнение компонентов пользовательских интерфейсов.

Одним из важных типов компонентов, который включается в библиотеку, является типовое ядро пользовательских интерфейсов. Этот вид компонентов нужен для погружения спроектированного пользовательского интерфейса на определенную платформу. Такие компоненты включают в себя набор необходимых рефепторных и эффекторных sc-операций и обеспечивает взаимодействие компонентов пользовательских интерфейсов между собой.

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

- просмотрщики содержимого sc-ссылок;
- редакторы содержимого sc-ссылок;
- трансляторы содержимого sc-ссылки в SC-код;
- трансляторы из SC-кода в содержимое sc-ссылок.

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

База знаний библиотеки компонентов включает в себя следующую информацию:
- спецификация компонентов;
- история версий компонентов;
- статистика использования компонентов.

Машина обработки знаний состоит из следующего набора операций:
- поиск компонента, по его спецификации;
- добавление нового компонента;
- удаление компонента;
- сравнение двух компонентов.

Пользовательский интерфейс библиотеки 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 компонентов.

Очевидным является то, что такая система должна быть реализована в интернет варианте, так чтобы она постоянно пополнялась и все имели доступ к самой актуальной версии компонентов.

В рамках библиотеки компонентов могут содержаться различные версии и модификации какого-либо компонента. К примеру, компонент просмотра sc.g-конструкций может иметь двумерную, трехмерную или же многослойную модификации, при этом каждая из модификаций компонента может иметь различные версии.

Использование библиотеки 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.

4. Интегрированные средства 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 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.

5. Методика 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.

Следующим этапом является создание задачно-ориентированной декомпозиции проектируемого пользовательского интерфейса. На этом этапе он 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 совместимых 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 компонентов, тем самым внося вклад в развитие технологии.

Таким образом, сама методика проектирования направлена на развитие технологии за счет разработки новых компонентов и пополнения ими библиотеки.

6. 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 компонентов пользовательских интерфейсов интеллектуальных систем. В этом разделе приводится полное формальное описание библиотеки 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

Описанные выше принципы и приемы позволяют проектировать интеллектуальные пользовательские интерфейсы для интеллектуальных систем, которые легко интегрируются в них и строятся на основе уже имеющихся компонентов. На основе предлагаемой технологии уже проектируются пользовательские интерфейсы некоторых прикладных систем. Что дает возможность говорить о работоспособности предлагаемого подхода. Все результаты, в том числе и исходные коды компонентов и ядра пользовательских интерфейсов, представлены на сайте проекта OSTIS.

Библиографический список

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

[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, 2011] Автоматизация разработки пользовательских интерфейсов с динамическими данными / В.В. Грибова, Н.Н. Черкезишвили // Материалы международной научно-технической конференции “Открытые семантические технологии проектирования интеллектуальных систем” - 2011. – с. 287-292

[Касьянов, 2003] Касьянов, В.Н. Графы в программировании: обработка, визуализация и применение/ В. Н. Касьянов, В. А. Евстигнеев // СПб.: BHV, 2003. – 1104 с.

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

[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.

[OSTIS, 2011] Открытая семантическая технология проектирования интеллектуальных систем [Электронный ресурс]. – 2011. - Режим доступа: http://ostis.net. – Дата доступа: 10.11.2011

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

SEMANTIC MODELS OF MULTIMODAL USER INTERFACES AND SEMANTIC THECHNOLOGY FOR THEIR DESIGN

Koronchik D. N.*

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

denis.koronchik@gmail.com

This article describes semantic technology for user interface design. This article contains description of semantic user interface model, wich is a main part of that technology.

Introduction

The development of software engineering environments has had a long and close relationship with the development of advanced user interface technologies. There are many technologies that allow to create user interfaces.

The only way to interact with machine is user interface. It would be impossible to interact with systems without it.

Main Part

The main purposes of semantic user interface technology are: to decrease time of user interface design by using components; lower level of developer qualification; to decrease requirements to user’s qualification; make user interface integration with intelligent system much simpler. The offered technology based on a four principles:

- unified semantic model lies in a basis of all designed user interfaces;
- user interfaces designed from components;
- design process supports by intelligent tools;
- technology includes help system, that helps developers in design process.

The offered technology includes: semantic model of user interfaces, components library, design tools, design technique, technique to study design process, help system for technology.

Conclusion

Principles and techniques, described in this paper, allows to create intelligent user, that designed by using components.