МЕТОДЫ АВТОМАТИЗИРОВАННОГО ПРОЕКТИРОВАНИЯ И СОПРОВОЖДЕНИЯ ПОЛЬЗОВАТЕЛЬСКИХ ИНТЕРФЕЙСОВ

Валерия Грибова

Аннотация: *В данной работе представлены методы автоматизации проектирования и сопровождения компонента представления информации в проекте пользовательского интерфейса. Для каждого метода описана его основная идея, область использования, проведено сравнение с аналогами.*

Ключевые слова: *Онтология, проект интерфейса, автоматическая генерация*

ACM Classification Keywords*: I.2.2 Artificial Intelligence – automatic programming*

Conference*: The paper is selected from XIV[th] International Conference "Knowledge-Dialogue-Solution" KDS 2008, Varna, Bulgaria, June-July 2008*

Введение

Разработка интерфейсов программных средств, удовлетворяющих требованиям пользователей, является одной из важнейших задач при создании программного обеспечения. Общей тенденцией является усложнение пользовательских интерфейсов, связанное как с увеличением функциональности программ, так и с различными, часто изменяющимися условиями их эксплуатации. В результате, его проектирование, разработка, модификация и сопровождение требуют значительных затрат времени, так как необходимо учитывать множество факторов, тесно взаимосвязанных между собой и зачастую противоречащих друг другу. Так, по оценкам специалистов, например [1], 50% времени, требуемого на разработку программного средства, уходит на пользовательский интерфейс; он занимает в среднем 48% программного кода.

Для снижения трудоемкости и времени разработки пользовательских интерфейсов предложен онтологический подход к автоматизации его разработки и сопровождения. Основная идея подхода заключается в формировании проекта интерфейса на основе предлагаемых разработчику онтологий и автоматическая генерация по проекту исполнимого кода интерфейса на некоторый язык программирования и связывание его с кодом прикладной программы (язык программирования и способ взаимодействия интерфейса с прикладной программой – локальный либо распределенный определяет разработчик). В результате разработка и сопровождение пользовательского интерфейса сводятся к разработке и сопровождению его проекта.

Опыт использования инструментария, основанного на онтологическом подходе показал, что разработка и сопровождение компонента представления информации проекта интерфейса при разработке сложных и комплексных интерфейсов остается трудоемкой, так требуют от разработчика знания принципов юзабилити[1], различных стандартов разработки, учета требований пользователей, среды использования интерфейса и др. Поэтому актуальными являются исследования, направленные на разработку методов и программных средств автоматизации проектирования компонента представления информации.

[1] Концепция разработки пользовательских интерфейсов, ориентированная на максимальное психологическое и эстетическое удобство для пользователя.

Проект пользовательского интерфейса

Проект пользовательского интерфейса задает множество процессов диалога с пользователем для осуществления ввода исходных данных в прикладную программу, вывода результатов ее работы, управления прикладной программой, а также для организации контекстно-зависимой помощи пользователю в процессе его взаимодействия с программным средством [2, 3]. Проект интерфейса состоит из следующих компонентов: системы понятий диалога, задач пользователя, представления информации, связи интерфейса с прикладной программой, сценария диалога, отображения.

Проект системы понятий диалога (ПСПД) описывает систему терминов предметной области, в которых выражаются входные/выходные данные прикладной программы, информация об управлении прикладной программой и ее пользовательским интерфейсом, а также об интеллектуальной поддержке действий пользователя. Данная компонента является конкретизацией Онтологии системы понятий диалога и представляет собой пару ПСДП=(G, σ), где G – ориентированный граф ПСПД, σ - разметка графа. Ориентированный граф ПСПД G=<Vertexes, Arcs, RootVertex>, где Vertexes - множество вершин графа, Arcs – множество дуг графа, RootVertex - корневая вершина. Множество вершин графа Vertexes=$\{Vertex_i\}_{i=1}^{vertexescount}$ состоит из двух подмножеств – множества терминальных и нетерминальных вершин, Vertexi $\in$Neterminal_Vertexes $\cup$ Terminal_Vertexes, где Neterminal_Vertexes - множество нетерминальных вершин, Terminal_Vertexes - множество терминальных вершин.

Множество дуг графа Arcs = $\{Arc_i\}_{i=0}^{arcscount}$, Arc_i=<$VertexFrom_i$, $VertexTo_i$ >, где $VertexFrom_i \in$ Vertexes, $VertexTo_i \in$ Vertexes. Корневой вершиной графа RootVertex является нетерминальная вершина, RootVertex$\in$Neterminal_Vertexes. Разметка графа σ - отображение множества вершин и дуг графа во множество имен Ω, где Ω = ИмяСистемыПонятий $\cup\{$ИмяГруппы$_i\}_{i=1}^{termgroupcount}$ $\cup$ $\{$ИмяТермина$_j\}_{j=0}^{termcount}$ $\cup$ $\{$ИмяАтрибута$_i\}_{i=1}^{attributecount}$ $\cup$ {Качественное значение$_i\}_{i=2}^{valuecount}$ $\cup\{$((ВещественноеМин, ВещественноеМакс), ВещественноеМера)$_i\}_{i=0}^{floatcount}$ $\cup\{$((ЦелоеМин, ЦелоеМакс), ЦелоеМера)$\}_{i=0}^{integercount}$ $\cup\{$Ni$\}_{i=0}^{stringscount}$ $\cup\{$Группа Терминов, Термин, Атрибут, Совместный, Несовместный, ВещественноеЗначение, ЦелоеЗначение, СтроковоеЗначение$\}$. Множество имен Ω определяется компонентами онтологии системы понятий диалога, при этом разметка графа определяется типом вершин $VertexFrom_i$ и $VertexTo_i$.

Проект задач пользователя (ПЗП). Любое программное средство предназначено для решения задач пользователя. Задачи могут быть разбиты на подзадачи, представляющие собой шаги для решения исходной задачи. Между подзадачами существуют отношения [4], которые определяют порядок и условия их выполнения. ПЗП=(T, σ), где T – дерево задач, σ - разметка дерева. Дерево задач T=<Vertexes, Arcs, RootVertex>. Разметка дерева σ - это отображение множества вершин дерева во множество имен Ω, где Ω =$\{$ИмяЗадачи$_i\}_{i=1}^{taskcount}$ $\cup$ ТипМножества. Множество имен Ω состоит из имен задач и имен множеств ТипМножества ={«выбор», «объединение», «разрешение», «деактивация»}. Разметка σ задается следующим образом: RootVertex$\rightarrow$ ИмяОбщейЗадачи, корневая вершина отображается в имя общей задачи. Каждая вершина $Vertex_i \rightarrow (Mark1_i, Mark2_i)$, где $Mark1_i \in$ $\{$ИмяЗадачи$_i\}_{i=1}^{taskcount}$, $Mark2_i \in$ТипМножества, (семантика множеств определена в [4]).

Проект представления информации описывает структуру и свойства визуального представления элементов WIMP[1]-интерфейсов и является конкретизацией модели онтологии WIMP-интерфейсов. Данный проект характеризуется множеством окон Windows, Windows=$\{Window_i\}_{i=1}^{windowcount}$, таких что

[1] Windows, Icons, Menus, Pointing devices

$Window_i \in Controls$ | Controltype = Окно-контейнер. Согласно модели онтологии WIMP-интерфейсов, каждый элемент интерфейса $Control_i$ задается своим типом, множеством параметров, функций и событий. Тип, функции и события элемента интерфейса $Control_i$ заданы в модели онтологии, соответственно, данный проект описывает значения параметров элементов интерфейса, характеризующих конкретный проект интерфейса. Параметрами окна $Window_i$ могут быть другие элементы интерфейса, в этом случае $Param\_{Typek} = Control_m$, где $Control_m \in Controls$.

Проект связи интерфейса с прикладной программой описывает способ взаимодействия интерфейса и прикладной программы, а также программные интерфейсы, посредством которых обеспечивается связь между интерфейсом и прикладной программой. Данный проект является конкретизацией соответствующей онтологии и представляет собой множество $Interfaces=\{Interface_i\}_{i=1}^{\text{int } erfacecount}$. Каждый элемент множества содержит описание программного интерфейса, предоставляемого прикладной программой, $Interface_i = < Interfacename_i, InteractionModel_i, Functions_i$, где $Interfacename_i$– уникальное имя программного интерфейса, $InteractionModel_i$ принимает значение из множества IModels={Локальная, Распределенная}, множество функций Functions - описание программных интерфейсов, предоставляемых прикладной программой.

Проект сценария диалога состоит из описания начального окна StartWindow, с которого начинается диалог с пользователем, $StartWindow=Window_i$, где $Window_i \in Windows$, а также множества возможных состояний States ={$State_i$}. Каждое состояние Statei содержит: описание события $Event_i$, множество переменных $Variables_i$, которые необходимы для описания последовательности инструкций $Instructions_i$, последовательность инструкций $Instructions_i=<Instruction_{ij}>_{j=1}^{instructioncount}$. $\}_{i=1}^{statecount}$. Инструкциями могут быть: вызовы программных интерфейсов прикладной программы, вызовы системных функций, составные элементы (условная конструкция, цикл), содержащие последовательности из перечисленных элементов). Проект сценария диалога является конкретизацией Онтологии сценария диалога.

Проект отображения в общем случае определяет множество отображений элементов одного из компонентов проекта интерфейса в другой: элементов проектов системы понятий диалога и задач пользователя в параметры элементов интерфейса проекта представления информации, параметров элемента интерфейса на выходные данные (результаты) прикладной программы проекта сценария диалога и др.

Методы построения проекта представления

Одной из задач при формировании проекта интерфейса является формирование проекта представления информации. Данный проект формируется по проектам системы понятий диалога и задач пользователя, т.е. разработчик интерфейса задает множество отображений указанных компонентов проекта в проект представления информации. Его задача сводится к тому, чтобы каждому термину предметной области, либо группе терминов сопоставить его (их) представление в интерфейсе – множество интерфейсных элементов с заданными свойствами (см. рис. 1).

При таком отображении возникают следующие проблемы:

- трудоемкость разработки проекта представления информации, если проект системы понятий диалога либо задач пользователя имеют большой размер (большая предметная область);

- сложность сопровождения проекта интерфейса: при изменении термина (терминов) предметной области необходимо также изменять проект представления информации;

- высокие требования к разработчику: знание им требований юзабилити, стандартов разработки, их совмещение с контекстом использования интерфейса, требованиями пользователей, предметной области к представлению информации и др.

Для реализации основного требования к проекту интерфейса – обеспечению его легкой модифицируемости и сопровождения предлагается несколько методов построения проекта представления информации. Выбор метода построения при разработке проекта интерфейса определяется его разработчиком и зависит от особенностей компонентов проекта (например, размера проекта системы понятий диалога, задач пользователя, особенностей предметной области к представлению информации и др.).

Метод прямого формирования проекта представления информации. В случае прямого формирования проекта представления информации значением параметра элемента интерфейса является прямая ссылка (см. рис. 1) либо на элемент проекта системы понятий диалога, либо на элемент проекта задач пользователя, т.е. для элемента интерфейса проекта представления WIMP-интерфейсов $Control_i \in Windows$, у которого $Param\_Type_{param}=String$, $Param\_Value_{param}=Значение$, где $Значение \in \{ИмяЗадачи_i\}_{i=1}^{taskcount} \cup ИмяСистемыПонятий \cup \{ИмяГруппы_i\}_{i=1}^{termgroupcount} \cup \{ИмяТермина_j\}_{j=0}^{termcount} \cup \{ИмяАтрибута_i\}_{i=1}^{attributecount} \cup \{Качественное\ значение_i\}_{i=2}^{valuecount}$. Например, на рис. 1, значениями параметра «Текст элемента» элементов пролистываемого списка является ссылка на значения атрибута «локализация» из проекта системы понятий диалога. При этом количество элементов пролистываемого списка определяется числом значений атрибута «локализация»; при изменении числа значений этого атрибута, изменяется количество элементов списка; аналогично, изменения самих значений этого атрибута в проекте системы понятий диалога приводят к изменениям параметра «Текст элемента».

Данный метод формирования проекта представления удобен, если проект системы понятий диалога имеет небольшой размер, а также требуется точное указание о представлении системы понятий диалога либо задач пользователя в интерфейсе (например, в случае особых запросов пользователей).

Метод регулярного формирования проекта представления информации. В данном случае значением параметра элемента интерфейса являются ссылки не на компоненты проекта системы понятий диалога, а на компоненты соответствующей онтологии (группы терминов, термины, значения, атрибуты), т.е. для элемента интерфейса проекта представления WIMP-интерфейсов $Control_i \rightarrow Element \mid Controli \in Windows, Element \in \{ГруппыТерминов, Термины, Атрибут, Качественные\ значения\}\}_{i=1}^{controlcount}$.

В результате такого формирования проекта представления каждая группа терминов (терминов, значений, атрибутов), входящая в проект системы понятий диалога, представляется выбранным дизайнером элементом интерфейса, при этом значением строкового параметра(ов) этого элемента интерфейса являются элементы Проекта системы понятий диалога.

Так, например, значению параметра «Элементы»= $\{Элемент\ списка_i\}_{i=1}^{elemcount}$ элемента интерфейса «Пролистываемый список» сопоставляются все качественные несовместные значения онтологии системы понятий диалога. Это означает, что все качественные совместные значения из проекта системы понятий диалога представляются в интерфейсе одинаковыми элементами интерфейса (см. рис. 2). При этом их количество может сколь угодно большим и определяется предметной областью.

Данный метод формирования проекта представления используется, если размер проекта системы понятий диалога имеет большой размер (сотни и тысячи понятий).

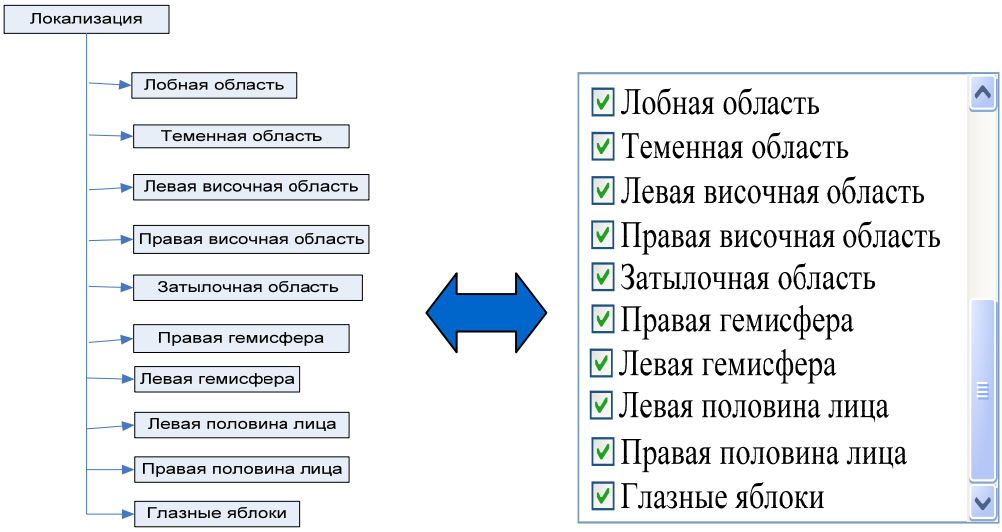

Рис. 1 Пример прямого формирования проекта представления информации

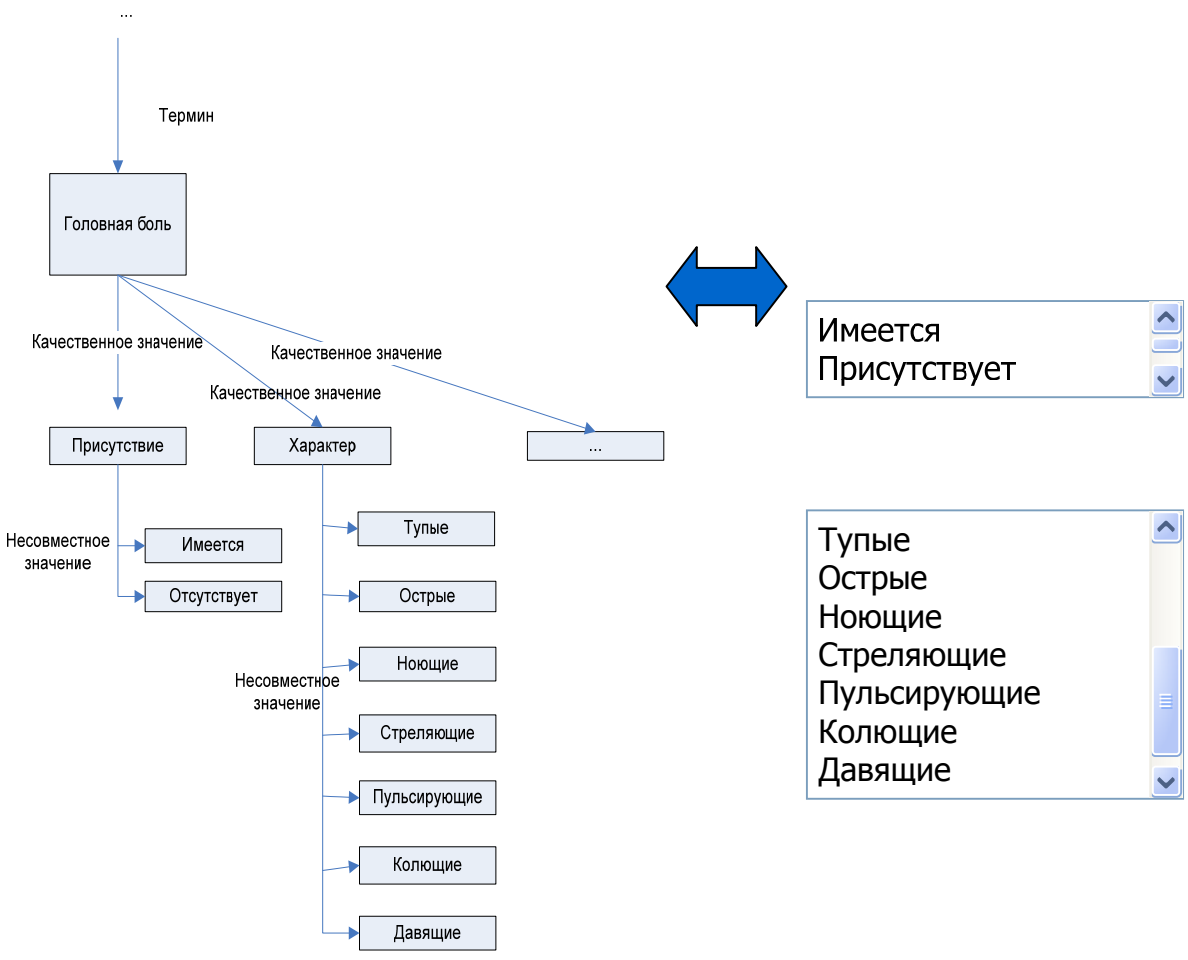

Рис. 2. Пример регулярного формирования проекта представления информации

Метод фрагментарного формирования проекта представления информации

При использовании данного метода формирования проекта представления информации, разработчик интерфейса разбивает проект системы понятий диалога на множество фрагментов, основываясь на требованиях пользователей, а также семантике и функциональности прикладной программы; каждому фрагменту проекта системы понятий диалога сопоставляются их представления в интерфейсе: $\{Fragment\_Presentastion \rightarrow Fragment\_Terms \mid Fragment\_Presentastion \in \{Control_i\}_{i=1}^{controlcount}, Control_i \in Windows, Fragment\_Terms \subset G\}_{i=1}^{fraggmentcount}$.

Основные положения метода заключаются в следующем:

- Каждому фрагменту проекта системы понятий диалога автоматически сопоставляется множество возможных представлений.
- Сопоставления осуществляются в соответствии с правилам юзабилити.
- Если для одного фрагмента возможно несколько представлений, не противоречащих правилам юзабилити, то либо:
  - право выбора единственного представления из множества допустимых предоставляется разработчику интерфейса;
  - представление автоматически выбирается в соответствии с системой умолчаний.
- Разработчику интерфейса предоставляется объяснение, содержащее протокол работы: описание набора критериев, по которым произведен выбор с указанием, какие из выбранных отображений были опровергнуты (не соответствуют правилам юзабилити и почему); какие критерии соответствуют правилам юзабилити и почему.
- Множество отображений должно расширяться (правила юзабилити и интерфейсные элементы постоянно развиваются уточняются, совершенствуются).

Для обеспечения расширяемости множества отображений разработана модель онтологии отображения представлений. В терминах онтологии описываются правила отображения элементов системы понятий диалога в их возможные представления в интерфейсе (множество интерфейсных элементов) с учетом правил юзабилити. Таким образом, формируется множество отображений (база знаний отображений). Любое отображение описывается парой – условием отображения и множеством возможных представлений. Условие отображения может состоять из множества условий, каждое из которых задает значения параметров, истинность которых проверяется в проекте системы понятий диалога. Параметры условий задаются в терминах онтологии системы понятий диалога. Например, в качестве условия может выступать количество качественных значений некоторого термина из проекта системы понятий диалога и способ их выбора (совместный либо несовместный). В зависимости от различных значений этих двух условий им сопоставляются различные представления: множество радио-кнопок, кнопок-флажков, раскрывающийся список, пролистываемый список с элементами различных типов и др. На рис. 3 представлены два различных его разбиения одного фрагмента проекта системы понятий диалога и соответствующий каждому разбиению фрагмент проекта представления информации.

Как видно из рис. 3, различные разбиения проекта системы понятий диалога приводят к различным проектам представления информации. В первом случае фрагмент имеет два разбиения, каждому из которых соответствует - окно («Боли» и «Повышение артериального давления»), во втором случае во фрагменте выделено одно разбиение, которому соответствует окно «Жалобы».

Данный метод формирования проекта представления также, как и предыдущий, удобен, если размер проекта системы понятий диалога имеет большой размер (сотни и тысячи понятий).

Рис. 3. Пример фрагментарного формирования проекта представления информации

Обсуждение результатов

В настоящее время рассмотренные выше методы реализованы и интегрированы в инструментарий для проектирования и сопровождения пользовательского интерфейса, основанный на онтологическом подходе. С использованием данных методов реализованы и сопровождаются интерфейсы для ряда программных средств.

Обсуждение полученных результатов целесообразно проводить с Моделеориентированными средствами (МОС) для разработки и автоматической генерации пользовательских интерфейсов [5-7], поскольку в Построителях интерфейсов такие методы не реализованы.

Метод прямого формирования проекта представления информации используется в некоторых МОС следующим образом: значениям параметров интерфейсных элементов присваиваются понятия предметной области либо задачи пользователя из соответствующих компонентов проекта интерфейса. Однако при их изменении приходится заново осуществлять присваивание, после чего производить перекомпиляцию интерфейса, в результате сильно возрастает трудоемкость его сопровождения. В данном методе вместо явного присваивания значений параметрам элементов интерфейса указываются ссылки на термины предметной области либо задачи пользователя, поэтому при их изменении автоматически происходят изменения и в проекте представления, перекомпиляция интерфейса при этом не требуется.

Метод регулярного формирования проекта интерфейса предложен впервые. Он позволяет значительно снизить трудоемкость разработки и сопровождения интерфейсов с большой системой понятий диалога. Например, проект системы понятий диалога для Системы интеллектуальной поддержки врача-уролога [8] состоит из 700 групп терминов и терминов и около 5000 вариантов значений. В этом случае метод, основанный на регулярном формировании проекта представления, позволяет очень быстро сформировать и автоматически, как и в предыдущем случае, сопровождать проект интерфейса при изменении системы понятий диалога.

Фрагментарные отображения используются в некоторых МОС. В этом случае разработчики определяют окно (множество окон) и информацию из модели задач либо системы понятий диалога, которая должна быть помещена в каждое окно. Далее автоматически выбираются представления для соответствующей информации. С одной стороны, такой подход уменьшает время разработки проекта представления информации, но с другой стороны, основная проблема, с которой столкнулись разработчики – сложность его сопровождения. Известно, что, во-первых, термины предметной области и задачи пользователя подвержены частым изменениям, во-вторых, любое автоматизированное представление требует последующего «ручного» пост-редактирования. В случае модифицирования терминов предметной области или задач пользователя все изменения, произведенные пост-редактированием, теряются. В результате сопровождение становится очень трудоемким. В некоторых МОС были предприняты попытки сохранять настройки пост-редактирования, однако, они не привели к положительному результату, так как не была решена основная проблема: что делать с настройками в случае появления новых терминов или задач, либо удаления существующих. В данной методе также, как и в в предыдущих, вместо явного присваивания значений параметрам элементов интерфейса указываются ссылки на термины предметной области либо задачи пользователя, при этом отображения производятся в соответствии с метриками юзабилити. Следующим недостатком данного метода, реализованного в некоторых МОС является их жесткая встроенность в инструментарий, в результате чего разработчик не знает правил, по которым проводятся автоматические преобразования, что также затрудняет и их расширение. В соответствии с методом, предложенным в данной работе, отображения явно указаны в базе отображений, которая формируется по соответствующей модели онтологии. Разработчику предоставляется возможность ее просмотра и расширения.

Благодарности

Работа выполнена при финансовой поддержке ДВО РАН в рамках Программы №15 фундаментальных исследований ОЭММПУ РАН "Проблемы анализа и синтеза интегрированных систем управления для сложных объектов, функционирующих в условиях неопределённости", проект "Синтез интеллектуальных систем управления базами знаний и базами данных при управлении сложными объектами в условиях неопределённости" и РФФИ, проект 06-07-89071-а "Исследование возможностей коллективного управления в семантическом вебе информационными ресурсами различных уровней общности".

Библиография

1. Myers Brad, Rosson Mary. Survey on user interface programming // Tech. Rpt. CMU-CS-92-113, Carnegie-Mellon, School of Comp. Sci., February 1992. http://citeseer.ist.psu.edu/myers92survey.html
2. Грибова В.В., Клещев А.С. Управление проектированием и реализацией пользовательского интерфейса на основе онтологий // Проблемы управления, 2006. №2. С.58-62.
3. Gribova V., Kleshchev A. From an ontology-oriented approach conception to user interface development International //Journal Information theories & applications. 2003. vol. 10, num.1, p. 87-94
4. Gribova V. Automatic generation of context-sensitive help using a user interface project // Proc. of XIIIth Intern. Conf. “Knowledge-dialogue-solution” – Varna, 2007. V.2. P. 417-422.
5. Puerta A. A model–based interface development environment IEEE Software, 14(1), July/August 1997. P. 41–47.
6. Puerta A.R. Issues in Automatic Generation of User Interfaces in Model-Based Systems. Computer-Aided Design of User Interfaces, ed. by Jean Vanderdonckt. Presses Universitaires de Namur, Namur, Belgium, 1996. P. 323-325.
7. Szekely P. Retrospective and Challenges for Model-Based Interface. 1996. http://citeseer.nj.nec.com/szekely96retrospective.html
8. Грибова В.В., Тарасов А.В., Черняховская М.Ю. Система интеллектуальной поддержки обследования больных, управляемая онтологией // Программные продукты и системы, 2007. №2. С. 49-51

Информация об авторе

Валерия Грибова – *к.т.н.**,** старший научный сотрудник отдела Интеллектуальных систем Института автоматики и процессов управления Дальневосточного отделения Российской академии наук, г. Владивосток, ул. Радио, 5, тел. +7 (4323) 314001, gribova@iacp.dvo.ru, http://www.iacp.dvo.ru/is.*