УДК 004.001;004.5

УПРАВЛЕНИЕ ПРОЕКТИРОВАНИЕМ И РЕАЛИЗАЦИЕЙ ПОЛЬЗОВАТЕЛЬСКОГО ИНТЕРФЕЙСА НА ОСНОВЕ ОНТОЛОГИЙ[1]

В. В. Грибова, А. С. Клещев

Институт автоматики и процессов управления, г. Владивосток

Предложен новый подход, учитывающий требования к разработке программных систем и базирующийся на анализе современного инструментария для разработки пользовательского интерфейса. Основная идея подхода — на основе онтологий, описывающих каждый компонент пользовательского интерфейса, сформировать информацию, необходимую для проектирования и реализации конкретного пользовательского интерфейса, и далее, по этой высокоуровневой спецификации автоматически сгенерировать исполнимый код интерфейса.

ВВЕДЕНИЕ

Усложнение функциональности программных систем и, как следствие, пользовательских интерфейсов, привело к появлению на рынке специальных инструментальных средств поддержки разработки пользовательских интерфейсов, каждое из которых основано на некоторой методологии анализа, проектирования, реализации и сопровождения. Такие средства должны обеспечивать [1—4]:

— поддержку проектирования и реализации различных типов диалога;

— инструментальную поддержку проектирования и автоматическую генерацию кода всех компонентов интерфейса;

— раздельную разработку и модификацию интерфейса и прикладной программы с последующим их связыванием;

— поддержку раздельной модификации различных компонентов пользовательского интерфейса;

— повторную используемость отдельных компонентов пользовательского интерфейса;

— поддержку оценивания проекта интерфейса;

— интеллектуальную поддержку разработчика, освобождающую его от изучения новых языков.

Кроме того, такие средства должны быть "открытыми" для внесения в них новых знаний о проектировании и реализации интерфейсов [5].

Существующие на рынке инструментальные средства, как правило, поддерживают диалог на основе экранных форм (с применением технологии WYSIWYG — What You See Is What You Get — и автоматической генерацией кода); ни одно средство не поддерживает проектирование и реализацию различных типов диалога.

Интегрированные пакеты, в состав которых входят построители интерфейсов, поддерживают проектирование, реализацию и повторное использование только визуального компонента интерфейса. Реализация остальных компонентов интерфейса осуществляется на языке программирования интегрированного пакета разработки — C^{++}, Паскаль и др. Остальным требованиям средства данной группы не удовлетворяют.

Системы управления пользовательским интерфейсом (User Interface Management Systems) [6, 7] поддерживают спецификацию интерфейса (каждая на своем языке) и автоматическую генерацию по ней некоторых компонентов интерфейса, а также раздельное проектирование пользовательского интерфейса и прикладной программы [8], однако остальным требованиям инструменты данной группы не удовлетворяют.

Средства разработки интерфейса, основанные на моделях (Model-Based Interface Development Environment) [9 — 12], поддерживают проектирование, модификацию, повторное использование и автоматическую генерацию кода всех компонентов интерфейса, а также раздельное проектирование и модификацию интерфейса и прикладной программы. Однако они требуют от разработчика изучения либо специальных декларативных языков для описания компонентов интерфейса (например, MIMIC), либо предлагают ему несколько известных формализмов (CORBA IDL, CTT, UML, Petri-net и др.), что снижает интеллектуальную поддержку разработчика [13]. Некоторые средства анализируют модели интерфейса, например, описанные в работах [14, 15]. Моделеориентированные средства не удовлетворяют требованию открытости, поэтому,

[1] Работа выполнена по программе № 16 ОЭММПУ РАН "Проблемы анализа и синтеза интегрированных технических и социальных систем управления", проект "Синтез интеллектуальных систем управления базами знаний и базами данных".

как замечено в работе [13], к тому времени, как такое средство выходит на рынок, оно требует модификации, по этой же причине устаревают и средства анализа.

Цель настоящей статьи состоит в представлении нового подхода к разработке интерфейса, в рамках которого могут быть созданы инструментальные средства, удовлетворяющие всем перечисленным выше требованиям.

1. ОСНОВНАЯ ИДЕЯ РАЗРАБОТКИ ПОЛЬЗОВАТЕЛЬСКОГО ИНТЕРФЕЙСА НА ОСНОВЕ ОНТОЛОГИЙ

Обзор специализированного инструментария показал, что наибольшему числу требований удовлетворяют средства для разработки интерфейса, основанные на моделях. Поэтому предлагаемый в настоящей работе подход является развитием моделеориентированного подхода. Новый подход также, как и моделеориентированный, исходит из раздельной разработки и модифицирования интерфейса и прикладной программы, разделения интерфейса на компоненты, необходимости инструментальной поддержки их проектирования, раздельной модификации, повторного использования, а также автоматической генерации кода интерфейса по модели.

Новым является определение модели пользовательского интерфейса, содержащей лишь ту информацию, которая может подвергнуться изменению в его жизненном цикле. Как показывает практика, проектирование интерфейса связано с четырьмя основными классами систем понятий:

— системой понятий пользователя, в терминах которой он осуществляет свое взаимодействие с прикладной программой;

— одной из систем понятий, в терминах которой определяются различные типы диалога;

— системой понятий для определения сценарий диалога;

— системой понятий, в терминах которой осуществляется связь между прикладной программой и пользовательским интерфейсом.

В интерфейсе можно выделить четыре компонента, каждый из которых определяется в терминах одной из этих систем понятий. В терминах этих же систем понятий может обеспечиваться интеллектуальная поддержка разработчика с помощью структурных и графических редакторов, а также определяются связи между компонентами.

Для обеспечения поддержки проектирования и реализации различных типов диалога разработчику интерфейса предлагаются три системы понятий, каждая из которых поддерживает проектирование одного из типов диалога — система понятий графического пользовательского интерфейса, поддерживающего разработку интерфейсов, основанных на формах или WIMP-интерфейсов (Windows, Icons, Menus, Pointing devices), система понятий графических статических сцен и система понятий для формирования текстов (в дальнейшем число систем таких понятий предполагается увеличить).

Для обеспечения "открытости" инструментария предлагаются следующие решения:

— явное представление выделенных систем понятий в форме моделей онтологий[2];

— предоставление разработчику структурных и графических редакторов для формирования разных компонентов модели интерфейса, управляемых моделями онтологий;

— предоставление специалистам, осуществляющим сопровождение инструментария, редакторов моделей онтологий и модели генерации кода, которая описывает соответствия между компонентами модели интерфейса и инструкциями целевого языка программирования.

Изменение моделей онтологий не приводит к модификации кода редакторов, которыми эти модели управляют, поскольку редакторы реализуются как интерпретаторы моделей онтологий. Если изменение онтологий требует изменений генератора кода интерфейса, то они осуществляются внесением изменений в модель генерации кода.

Для обеспечения открытости средств оценивания проекта интерфейса эти средства модели должны управляться базой знаний о дефектах. База знаний формируется и модифицируется с помощью редактора базы знаний о дефектах, управляемого моделью онтологии дефектов интерфейса, и, соответственно, является расширяемой (модифицируемой).

[2] *Онтология* — явное описание концептуализации. Она может иметь различные формы, но обязательно включает в себя словарь терминов, спецификацию их смысла, а также описание связей между терминами [16].

Повторная используемость компонентов интерфейса обеспечивается многоуровневыми библиотеками повторно используемых компонентов.

Таким образом, модель интерфейса проектируется на основе универсальных моделей онтологий для формирования каждого компонента модели интерфейса. В интерфейсе можно выделить следующие универсальные модели онтологий: $\boldsymbol{D}$ — модель онтологии пользователя; $\boldsymbol{G}$ — модель онтологии представления, которая состоит из трех моделей онтологий, определяющих один из типов диалога (G_1, G_2, G_3); $\boldsymbol{S}$ — модель онтологии сценария диалога, $\boldsymbol{L}$ — модель онтологии связи с прикладной программой. Каждая из моделей онтологий имеет вид: $<\boldsymbol{Name}, \boldsymbol{OS}>$, где $\boldsymbol{Name}$ — множество имен, $\boldsymbol{Name} = \{<\boldsymbol{n}, \{\boldsymbol{sn}\}>\}$, $\boldsymbol{n}$ — имя термина, $\boldsymbol{sn}$ — характеристика (атрибут) этого термина. $\boldsymbol{OS}$ — множество онтологических соотношений. Формирование компонента модели интерфейса сводится к заданию значений понятий соответствующей универсальной модели онтологии. Соответственно, модель интерфейса есть $\{\boldsymbol{D}', \boldsymbol{G}', \boldsymbol{L}', \boldsymbol{S}'\}$.

2. ПРОБЛЕМНО-НЕЗАВИСИМЫЕ МОДЕЛИ ОНТОЛОГИЙ

В общем случае каждая из проблемно-независимых моделей онтологий может быть представлена графом с размеченными вершинами и дугами $<C, L>$, в котором имеется несколько начальных вершин (за исключением модели онтологии сценария диалога, которая может содержать только одну начальную вершину). Вершины графа C соответствуют терминам онтологии, дуги графа L — отношениям между ними. С каждой вершиной графа связан термин и множество его свойств. Дуги графа снабжены разметкой, состоящей из имени и кардинальности отношения, которая задается либо в виде неотрицательного целого числа, либо в виде интервала на множестве неотрицательных целых чисел.

Модель онтологии пользователя определяет четыре класса терминов: группы, понятия, характеристики и области возможных значений (ОВЗ). Группа — это принятое в рассматриваемой предметной области объединение концептуально связанных групп и (или) понятий. Понятия делятся на простые, не имеющие внутренней структуры, и составные. Составные понятия обладают набором

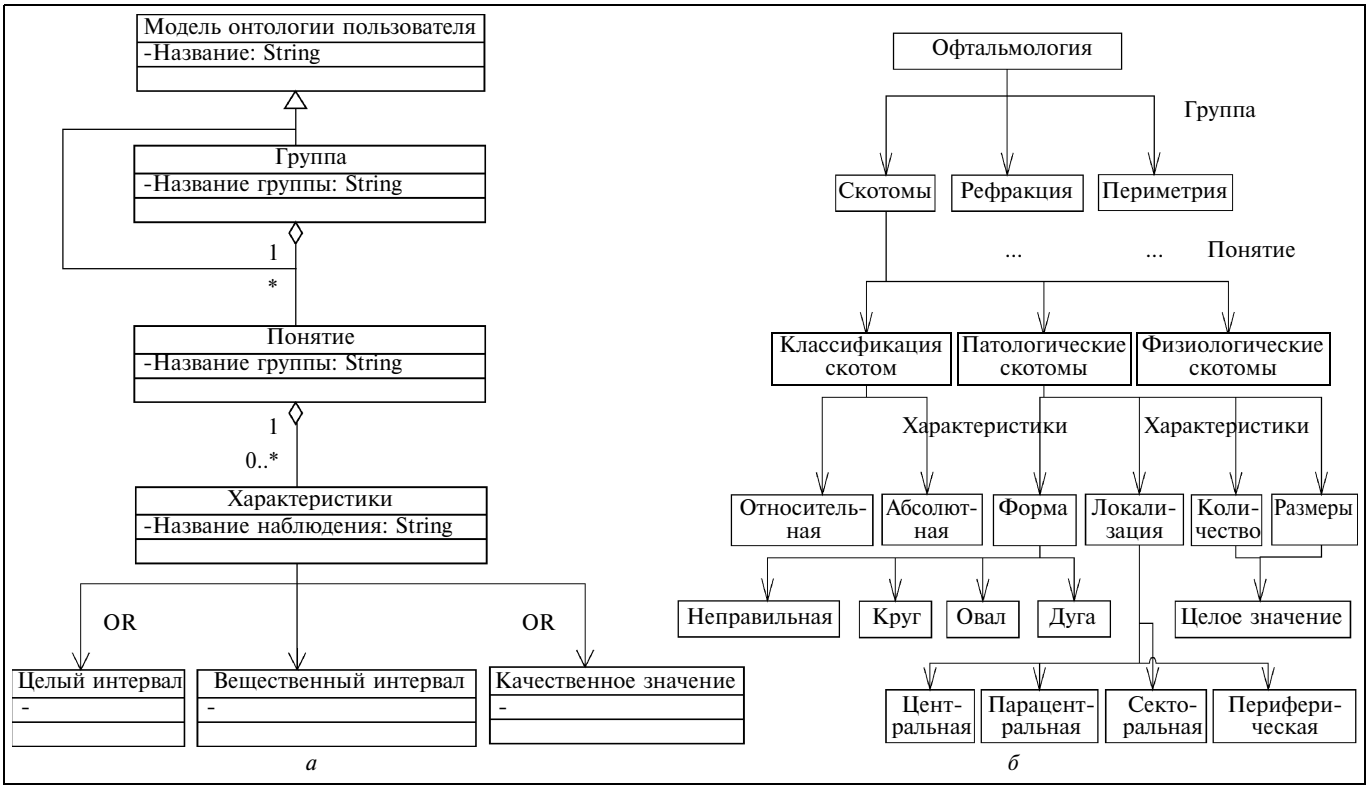

Рис. 1. Модель системы понятий пользователя:

а — универсальная модель онтологии; *б* — модель системы понятий пользователя для раздела медицины “офтальмология” в терминах универсальной модели онтологии

характеристик. Характеристика также может быть простой или составной; в последнем случае она обладает набором характеристик. Каждая простая характеристика или простое понятие описываются ОВЗ либо качественных, перечисляемых в описании ОВЗ, либо числовых (в этом случае ОВЗ задается диапазоном значений). Качественные значения могут быть “взаимоисключающими” или “совместными”.

На рис. 1 представлена модель онтологии пользователя средствами языка UML [17], а также сформированный на ее основе фрагмент системы понятий для раздела медицины “офтальмология”.

Модель онтологии представления состоит из трех моделей онтологий — модели онтологии графического пользовательского интерфейса (ГПИ), модели онтологии графических статических сцен на плоскости, модели онтологии для формирования текстов. Каждая из моделей онтологий позволяет реализовать один из типов диалога.

Модель онтологии ГПИ предназначена для формирования компонента представления информации на основе экранных форм. Она определяет две основные группы элементов — окна и оконные элементы управления, а также три дополнительные группы — панели управления, оконные меню и вспомогательные средства.

Модель онтологии графических статических сцен на плоскости определяет базовое графическое изображение, наполнения и примитивы. Базовое графическое изображение — это произвольный графический рисунок, схема, эскиз и т. п., являющееся основой для нанесения на него различных изображений. Базовое графическое изображение определяется своим именем, изображением, типом и соответствующим типу описанием его составляющих. Наполнение задает возможные варианты цветового и рельефного изображения базы. Для каждого базового изображения может быть задано множество возможных наполнений, которые определяются именем наполнения, его цветом и текстурой. Примитив задает возможные варианты изображений, наносимых на базовое изображение. Для каждой базы может быть задано множество возможных примитивов. Каждый примитив определяется именем, типом и соответствующим типу описанием. Тип примитива может быть либо предопределенным, либо строящимся, либо сложным. У предопределенных примитивов изображение определено заранее. Для строящегося примитива необходимо задать имя, форму, цвет линии и фона. Сложный примитив — это совокупность строящихся примитивов, соединенных между собой линиями заданного цвета и типа.

Модель онтологии для формирования текстов определяет конструкции для описания структуры и способа порождения текста на основе результатов (выходных данных прикладной программы).

Конструкции для описания структуры и способа порождения текста состоят из последовательности элементов описания, каждый из которых определяет зависимость порождаемого текста от выходных данных прикладной программы. Конструкции онтологии для формального описания структуры и способа порождения текста: альтернатива, цикл, выводимое множество и строка.

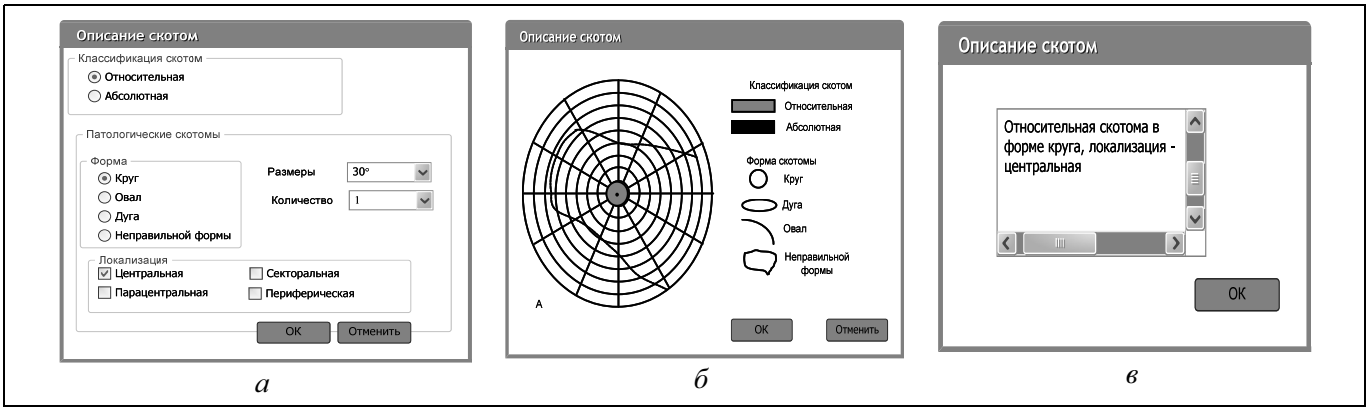

Рис. 2. Представление информации из раздела медицины "офтальмология" в различных формах:
а — интерфейс, основанный на формах; *б* — интерфейс, основанный на графических сценах; *в* — интерфейс в виде текста

Цикл и альтернатива, в свою очередь, также содержат элементы описания текста в качестве компонентов.

На рис. 2 показаны экранные формы, соответствующие трем типам диалога для фрагмента модели системы понятий пользователя, представленной на рис. 1. Экранные формы проектируются с помощью описанных выше моделей онтологий. Так, рис. 2, *а* соответствует типу диалога, основанному на экранных формах, рис. 2, *б* — типу диалога, основанному на графических сценах. В этом случае заполнение результатов осмотра больного включает формы с изображением схемы полей зрения и множеством условных обозначений, которые наносятся на эти формы и служат результатами осмотра. На рис. 2, *в* приведен фрагмент текста, описывающий те же данные и сформированный на основе модели онтологии для формирования текстов.

Модель онтологии сценария диалога определяет абстрактные термины для описания реакций на события (наборы действий, выполняемых при возникновении событий, источники событий, вид режимов переходов между окнами, способы выбора экземпляров окна и др.). Вершины графа данной модели онтологии соответствуют оконным классам модели онтологии графического пользовательского интерфейса, дуги графа — переходам между вершинами. Каждая вершина описывается именем класса модели онтологии графического пользовательского интерфейса, именем окна, множественностью, а также реакцией окна на события. Различаются три типа множественности: единичная, конечная и произвольная. Реакция окна на события — это набор действий, выполняемых при возникновении различных событий. Каждая реакция описывается тройкой: источником событий (экземпляром оконного класса, прикладной программой, оконным элементом управления), именем события, принадлежащего источнику, и реакцией на событие. Каждая дуга описывает режим перехода между окнами, способ выбора нового окна, а также событие, которое инициирует переход и вызов функций прикладной программы. Любой из трех типов диалогов использует оконные классы модели онтологии ГПИ, что позволяет описать сценарий диалога, соответствующий различным типам диалога.

Модель онтологии связи с прикладной программой определяет множество программных интерфейсов, предоставляемых прикладной программой. Каждый программный интерфейс реализует модель взаимодействия с программой, которая может быть локальной или распределенной, а также список функций, предоставляемых данным интерфейсом. Каждая функция характеризуется типом возвращаемого значения и множеством параметров, также определяемых типом.

3. УПРАВЛЕНИЕ ПРОЦЕССОМ ПРОЕКТИРОВАНИЯ И ГЕНЕРАЦИЕЙ ПРОГРАММНОГО КОДА НА ОСНОВЕ ОНТОЛОГИЙ

Управление проектированием пользовательского интерфейса осуществляется разработчиком пользо-

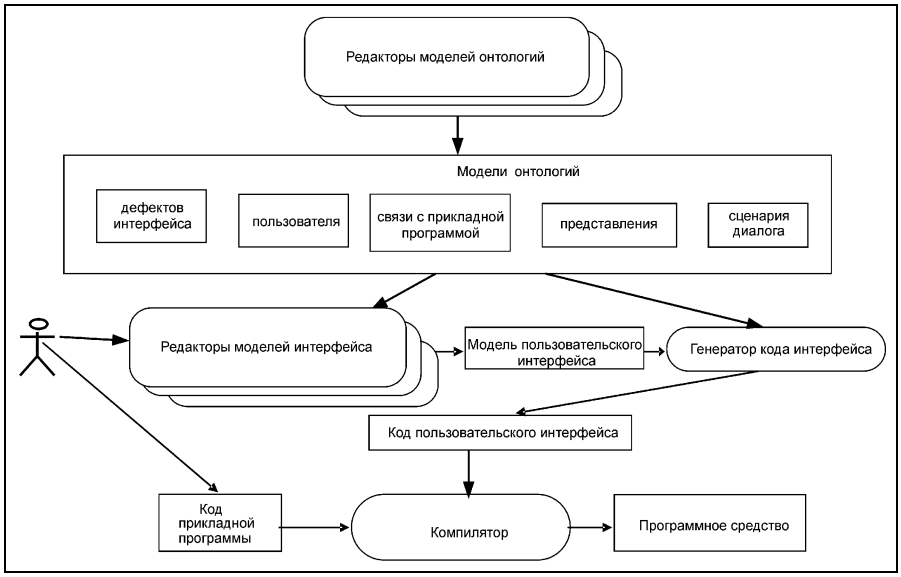

Рис. 3. Базовая архитектура инструментального комплекса для разработки пользовательского интерфейса

вательского интерфейса, а модели онтологий, которые управляют структурными и графическими редакторами, регламентируют набор правил для задания понятий, их характеристик и значений, определяющих специфику каждого компонента пользовательского интерфейса. На рис. 3 изображена базовая архитектура инструментального комплекса.

Генератор кода каждому компоненту модели онтологии сопоставляет набор исходных кодов на некотором языке программирования и связывает их вместе. Кроме того, генератор вносит в программный код дополнительные конструкции, предоставляющие возможность редактирования полученного кода в одном из существующих средств разработки для данного языка программирования, а также коды программных интерфейсов, структур данных и организации взаимодействия (набора инструкций связывания интерфейса и прикладной программы). В настоящее время разработан прототип инструментального средства, который поддерживает генерацию кода на языках C# и Java.

ЗАКЛЮЧЕНИЕ

Предложен подход к разработке пользовательского интерфейса, удовлетворяющий всем современным требованиям к таким средствам.

Для поддержки проектирования и реализации различных типов диалога предложены три модели онтологии — модель онтологии графического пользовательского интерфейса, графических статических сцен на плоскости, модель онтологии для формирования текстов. В результате, одной модели системы понятий пользователя и модели связи с прикладной программой может соответствовать несколько типов диалога, основанных на экранных формах, графических сценах и текстах. Для расширения типов реализуемых диалогов разработана модель онтологии графических динамических сцен на плоскости, разрабатывается редактор таких сцен, модель и методы автоматической генерации диалогов, основанных на динамических сценах.

Модель интерфейса состоит из следующих компонентов: модели системы понятий пользователя, модели представления, модели сценария диалога и модели связи с прикладной программой. Инструментальная поддержка проектирования обеспечивается набором редакторов для проектирования. Сформированная с их помощью модель затем автоматически преобразуется в исполнимый код на некоторый целевой язык программирования.

Реализация предложенного подхода должна обладать следующими особенностями:

- инструментальный комплекс поддерживает раздельную разработку и модификацию интерфейса и прикладной программы, а также связывание кода отдельно разработанных компонентов;
- повторная используемость обеспечивается многоуровневыми библиотеками компонентов: библиотеками шаблонов и библиотеками представлений систем понятий пользователя;
- интеллектуальная поддержка разработчика осуществляется на основе проблемно-независимых моделей онтологий, отражающих специфику каждого компонента модели, а также структурных и графических редакторов, управляемых этими моделями онтологий и освобождающих, таким образом, разработчиков от изучения различных языков для формирования компонентов модели;
- средства оценивания поддерживаются и управляются базой знаний о дефектах интерфейса, в настоящее время описывающей более 100 дефектов интерфейса;
- для обеспечения открытости инструментария графические и структурные редакторы реализуются как интерпретаторы моделей онтологий; для их модификации предлагаются редакторы моделей онтологий, это позволяет изменять онтологии без изменения кода; если модифицирование инструментария требует изменения кода интерфейса, разработчик вносит их в модель генерации кода, которая описывает соответствия между компонентами модели интерфейса и инструкциями целевого языка программирования.

ЛИТЕРАТУРА

1. *Schlungbaum E.* Model-based User Interface Software Tools Current state of declarative models. — 1996. <http://www.isys.ucl.ac.be/bchi/cadui/96/files96/Schlungbaum-CADUI96.pdf>.
2. *Гультяев А. К., Машин В. А.* Проектирование и дизайн пользовательского интерфейса. — СПб.: КОРОНА принт, 2000. — 352 с.
3. *Szekely P.* Retrospective and Challenges for Model-Based Interface. — 1996. <http://citeseer.nj.nec.com/szekely96retrospective.html>.
4. *Ivory M. Y., Hearst M. A.* State of the Art in Automating Usability Evaluation of User Interfaces // ACM Computing Surveys. — 2001. — Vol. 33, N 4. — P. 1—47.
5. *Puerta A. R.* Issues in Automatic Generation of User Interfaces in Model-Based Systems. — 1996. <http://smi-web.stanford.edu/projects/mecano/publicat.htm>.
6. *Singh G., Green M.* A High-level User Interface Management System // Proc. of SIGCHI'89, 1989. — P. 133—138.
7. *Lowgren J.* Knowledge-Based Design Support and Discourse Management in User Interface Management Systems // Linkoping Studies in Science and Technology / Dissertations N 239. — 1989. — 224 p.
8. *Myers B. A.* User Interface Software Tools // ACM Trans. on Computer-Human Interaction. — 1995. — Vol. 2, N 1. — P. 64—103.
9. *Szekely P., Sukaviriya P., Castells P., Muthukumarasamy J., Salcher E.* Declarative Interface Models for User Interface Construction Tools: the Mastermind Approach // Engineering for Human-Computer Interaction. — 1996. <http://www.isi.edu/isd/Mastermind/mastermind-ia.htm>.
10. *Puerta A.* Supporting User — Centered Design of Adaptive User Interfaces via Interface Models // Proc. of First Annual Workshop On Real-Time Intelligent User Interfaces for Decision Support and Information Visualization. — 1998. <http://smi-web.stanford.edu/projects/mecano/publicat.htm>.
11. *Wiecha C., Bennett W., Boies S., Gould J., Green S.* ITS: A Tool for rapidly Developing Interactive Applications // ACM Trans. on Information Systems. — 1990. — Vol. 8, N 3. — P. 204—236.
12. *Da Silva P., Griffiths T., Paton N.* Generating User Interface Code in a Model-Based User Interface Development Environment // Proc. of Advanced Visual Interfaces, ACM Press, 2000. — P. 155—160.
13. *Puerta A.* 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.
14. *Bodart F., Hennebert A., Leheureux J., Vanderdonckt J.* Computer-Aided Window Identification in Trident // Proc. of INTERACT'95. — London: Chapman & Hall, 1995. — P. 331—336.
15. *Fischer G., Nakakoji K., Ostwald J., Stahl G., Sumner T.* Embedding Computer-Based Critics in the Context of Design // Proc. of INTERCHI'93. — New-York: ACM Press, 1993. — P. 157—164.
16. *Uschold M.* Knowledge Level Modeling: Concepts and Terminology // The Knowledge Engineering Review. — 1998. — Vol. 13, N 1. — P. 5—29.
17. *The unified* modeling language. <http://www.uml.org>.

☎ *(4232) 31-40-01*

E-mail: gribova@iacp.dvo.ru, kleschev@iacp.dvo.rui ❑