На правах рукописи

Тарасов Алексей Владимирович

РАЗРАБОТКА И ИССЛЕДОВАНИЕ МЕТОДОВ ГЕНЕРАЦИИ И СОПРОВОЖДЕНИЯ WIMP-ИНТЕРФЕЙСОВ

05.13.11 – математическое и программное обеспечение вычислительных машин, комплексов и компьютерных сетей

Автореферат
диссертации на соискание ученой степени
кандидата технических наук

Владивосток – 2007

Работа выполнена в Отделе интеллектуальных систем Института автоматики и процессов управления Дальневосточного отделения РАН.

Научный руководитель: кандидат технических наук,
старший научный сотрудник
Грибова Валерия Викторовна

Официальные оппоненты: доктор технических наук,
старший научный сотрудник
Бобков Валерий Александрович

кандидат технический наук,
доцент Шевченко Игорь Иванович

Ведущая организация: Институт систем информатики
Сибирского отделения РАН (г. Новосибирск)

Защита состоится 19 октября 2007 г. в 10:00 часов на заседании диссертационного совета Д 005.007.01 в Институте автоматики и процессов управления Дальневосточного отделения РАН по адресу: 690041, г. Владивосток, ул. Радио, 5.

С диссертацией можно ознакомиться в библиотеке Института автоматики и процессов управления Дальневосточного отделения РАН.

Автореферат разослан "___" _____ 2007 г.

Ученый секретарь
диссертационного совета Д 005.007.01, к.т.н. Лебедев А.В.

ОБЩАЯ ХАРАКТЕРИСТИКА РАБОТЫ

Актуальность темы. 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 и часто изменяющимися условиями их эксплуатации. В результате на пользовательский интерфейс уходит существенная часть общих затрат на разработку программного средства. Поэтому исследования в области человеко-машинных интерфейсов, направленные на уменьшение трудоемкости разработки и сопровождения являются актуальными. Значительный вклад в решение этой проблемы внесли российские и зарубежные ученые: Гаврилова Т.А., Соснин П.И., Хорошевский В.Ф., Myers B.A., Puerta A.R., Szekely P.A. и другие.

В настоящее время наибольшее распространение получили WIMP[1]-интерфейсы, в которых взаимодействие с пользователем производится с помощью различных элементов интерфейса (кнопок, меню, списков, полей ввода и т.д.), предназначенных для ввода, вывода и управления информацией. Число таких элементов постоянно увеличивается, а их структура и поведение усложняются. При разработке пользовательских WIMP-интерфейсов в настоящее время используется два подхода, поддерживаемых специализированным инструментарием. Для каждого подхода существует свой класс таких инструментов – построители интерфейса (Interface Builders) или моделеориентированные средства (MB IDE – Model Based Interface Development Environments). Построители интерфейса предназначены для автоматизации процесса разработки их визуального представления с помощью структурных и визуальных редакторов. С использованием построителей интерфейса разрабатывается большинство современных WIMP-интерфейсов. Моделеориентированные средства (МОС) основаны на принципе раздельного проектирования и реализации интерфейса с прикладной программой с последующим их связыванием и предназначены для автоматической генерации кода пользовательского интерфейса по его модели. При этом модель интерфейса разбивается на несколько компонентов и предназначена для описания всех аспектов взаимодействия с пользователем.

Однако в каждом классе инструментов остался нерешенным ряд проблем, в результате которых трудоёмкость разработки и сопровождения интерфейсов остаётся высокой. Так в построителях интерфейса лишь визуальное представление и некоторые компоненты сценария диалога могут описываться с использованием структурных и визуальных редакторов, все остальные компоненты должны

[1] Windows, Icons, Menus, Pointing devices

описываться на языке программирования. В МОС для описания некоторых компонентов модели интерфейса требуется изучение специализированных языков, что увеличивает трудоемкость разработки. Построители интерфейса и МОС ограничены одной платформой и языком программирования, на которые генерируется код интерфейса. Поддержка организации взаимодействия (локального или сетевого) с прикладной программой либо ограничена каким-либо одним фиксированным способом взаимодействия (в МОС), либо полностью отсутствует (в построителях интерфейса разработчик вручную программирует этот компонент программного средства). Разработка и особенно сопровождение интерфейсов с большой системой понятий диалога с помощью указанного инструментария имеют высокую трудоемкость, поскольку в построителях интерфейса отсутствует механизм поддержки отдельного (от компонента представления) описания терминов предметной области; а в МОС, несмотря на наличие специальных возможностей для описания терминов предметной области и генерации пользовательских интерфейсов на их основе, возникает необходимость заново генерировать интерфейс и формировать новую версию программного средства при любом изменении структуры терминов или связей между ними. В построителях интерфейса и МОС возможность повторного использования компонентов пользовательского интерфейса ограничена копированием элементов интерфейса из уже разработанных интерфейсов.

Все вышесказанное определяет актуальность исследований, направленных на решение проблемы автоматизации разработки и сопровождения WIMP-интерфейсов.

Диссертационная работа является частью исследований, выполняемых в рамках предложенного в Институте автоматики и процессов управления (ИАПУ) ДВО РАН *онтологического подхода* к автоматизации разработки и сопровождения пользовательских интерфейсов. Основная идея подхода заключается в (1) разбиении интерфейса на компоненты в соответствии с системами понятий различных групп специалистов, осуществляющих его разработку и сопровождение; (2) построении для каждой системы понятий модели онтологии, в терминах которой разработчики интерфейса формируют соответствующий компонент его модели; (3) автоматической генерации кода пользовательского интерфейса по его модели. В работах Клещева А.С. и Грибовой В.В. разработана концепция автоматизации проектирования, реализации и сопровождения пользовательского интерфейса на основе онтологического подхода; определены компоненты модели интерфейса и их структура; а также модели онтологий системы понятий диалога, связи интерфейса с прикладной программой и сценария диалога. Но не были разработаны онтология WIMP-интерфейсов, методы формирования модели и методы генерации по ней исходного кода WIMP-интерфейсов.

Целью диссертационной работы является разработка и исследование моделей, методов и инструментального средства для генерации и сопровождения WIMP-интерфейсов на основе онтологического подхода.

Для достижения поставленной цели необходимо решить следующие **задачи:**

1) Разработать модель онтологии WIMP-интерфейсов.
2) Разработать метод формирования модели WIMP-интерфейса.
3) Разработать метод генерации исходного кода WIMP-интерфейса по его модели.
4) Разработать методы реализации инструментального средства для проектирования, автоматической генерации и сопровождения WIMP-интерфейсов.
5) Разработать технологию проектирования и сопровождения WIMP-интерфейсов с помощью разработанного инструментального средства и выполнить ее практическую проверку при разработке WIMP-интерфейсов программных систем.

Методы исследования. Для решения указанных задач использовались методы искусственного интеллекта, теория множеств, теория формальных языков, методы объектно-ориентированного проектирования, методы системного программирования.

Научная новизна работы состоит в следующем:

- впервые разработана модель онтологии WIMP-интерфейсов;

- предложен метод формирования модели WIMP-интерфейса в терминах онтологий (а не языков спецификаций);

- разработан абстрактный язык для описания структуры исходного кода WIMP-интерфейсов, не зависящий от конкретной платформы и языка его реализации;

- метод генерации исходного кода WIMP-интерфейса ориентирован на абстрактный (а не конкретный) язык.

Практическая ценность работы заключается:

- в создании инструментального средства (Onto Dev) для разработки, автоматической генерации и сопровождения WIMP-интерфейсов на основе онтологий;

- в разработке с помощью Onto Dev WIMP-интерфейса Системы интеллектуальной поддержки обследования больных для врача-уролога;

- в разработке с помощью Onto Dev WIMP-интерфейса Системы анализа газетных объявлений о купле-продаже недвижимости;

- в использовании Onto Dev при выполнении практических занятий по дисциплине «человеко-машинный интерфейс», курсовых и дипломных работ студентами Института математики и компьютерных наук Дальневосточного государственного университета.

Апробация работы. Основные положения диссертации докладывались и обсуждались на Дальневосточных математических школах-семинарах имени академика Е.В. Золотова (Владивосток, 2002-2004; 2007), Первой международной

конференции «Системный анализ и информационные технологии» (Переславль-Залесский, 2005), Международной научной конференции «Интеллектуальные и многопроцессорные системы» (Таганрог, 2005), Открытом дальневосточном конкурсе студентов, аспирантов и молодых специалистов «Программист-2006» (Владивосток, 2006), Научной сессии МИФИ (Москва, 2006), Международной конференции «Параллельные вычисления и задачи управления» (Москва, 2006), на семинарах отдела интеллектуальных систем ИАПУ ДВО РАН и базовой кафедры программного обеспечения ЭВМ ДВГУ (Владивосток, 2002-2007).

Публикация результатов работы. По материалам диссертации опубликовано 17 печатных работ и тезисов докладов конференций, в том числе в двух журналах, рекомендуемых ВАК РФ для опубликования научных результатов диссертаций.

Структура и объем работы. Диссертационная работа состоит из введения, пяти глав и заключения, изложенных на 138 страницах, списка литературы, включающего 141 наименование, и двух приложений. Диссертация содержит 44 рисунка.

СОДЕРЖАНИЕ РАБОТЫ

Первая глава содержит обзор литературы. В ней вводятся основные понятия WIMP-интерфейсов, анализируются существующие средства их разработки: построители интерфейса и МОС. Рассматриваются возможности этих средств при разработке интерфейсов для различных платформ и языков программирования, методы организации как локального, так и сетевого взаимодействия интерфейса с прикладной программой; механизмы повторного использования компонентов пользовательского интерфейса, сопровождения интерфейса и расширения инструментария. Рассматривается онтологический подход к разработке пользовательских интерфейсов, описанный в работах Грибовой В.В. и Клещёва А.С.

Во второй главе описывается модель онтологии WIMP-интерфейсов, которая состоит из двух уровней: метаонтологии, предназначенной для описания структуры элементов интерфейса и непосредственно онтологии, содержащей описание множества элементов интерфейса в соответствии со структурой, представленной в метаонтологии.

Модель метаонтологии WIMP-интерфейсов характеризуется множеством элементов интерфейса, $\mathrm{UIElements}=\{\mathrm{UIElement}_i\}_{i=1}^{controlscount}$.

Каждый элемент интерфейса $\mathrm{UIElement}_i=<\mathrm{UIElementType}_i, \mathrm{Parameters}_i, \mathrm{Events}_i, \mathrm{Functions}_i>$, где $\mathrm{UIElementType}_i$ – тип элемента интерфейса, $\mathrm{Parameters}_i$ – множество параметров, $\mathrm{Parameters}_i=\{\mathrm{Param}_{ij}\}_{j=0}^{paramcount}$, Events_i – множество событий, $\mathrm{Events}_i=\{\mathrm{Event}_{ij}\}_{j=0}^{eventcount}$, $\mathrm{Functions}_i$ – множество функций, $\mathrm{Functions}_i=\{\mathrm{Function}_{ij}\}_{j=0}^{funccount}$.

В результате анализа WIMP-интерфейсов были выделены следующие типы элементов интерфейса, $UIElementType_i \in$ {Окно-контейнер, Элемент управления, Вспомогательный элемент}.

Элементы интерфейса типа «Окно-контейнер» (окна) предназначены для группировки элементов интерфейса в семантически связанные группы. Элементы интерфейса типа «Элемент управления» (элементы управления) предназначены для операций ввода/вывода данных и вызова команд. Элементы интерфейса типа «Вспомогательный элемент» используются в качестве компонентов окон и элементов управления.

Каждый параметр элемента интерфейса $Param_i$ описывается своим типом и значением, т.е. $Param_i$=<$Param\_Value_i$, $Param\_Type_i$>, где $Param\_Value_i$ – значение параметра, $Param\_Type_i$ - тип параметра.

Тип параметра элемента интерфейса $Param\_Type_i \subset Param\_Type$, где Param_Type = String $\cup$ Integer $\cup$ Float $\cup$ Boolean $\cup$ Image $\cup$ Enumeration $\cup$ UIElem, UIElem $\in$ UIElements, Param_Value $\in$ Param_Type.

События элемента интерфейса $Events_i=\{Event_{ij}\}_{j=0}^{eventcount}$ задают множество тех событий, на которые элемент интерфейса может реагировать.

Каждое событие $Event_{ij}$ описывается следующим образом: $Event_{ij}$=<$Event\_Name_{ij}$, $Event\_Parameters_{ij}$>. $Event\_Name_{ij}$ – имя события, $Event\_Parameters_{ij}$ – множество параметров события.

Множество $Event\_Parameters_{ij} = \{Event\_Parameter_{ijk}\}_{k=0}^{eventparametercount}$. Каждый параметр задается своим именем и типом: $Event\_Parameter_{ijk}$=<$Event\_Param\_Name_{ijk}$, $Event\_Param\_Type_{ijk}$>, где $Event\_Param\_Name_{ijk}$ – уникальное имя параметра, $Event\_Param\_Type_{ijk}$ - тип параметра, $Event\_Param\_Type_{ijk} \in$ Func_Type (описание Func_Type см. ниже).

Функции элемента интерфейса $Functions_i=\{Function_{ij}\}_{j=0}^{funccount}$ описывают множество возможных действий, которые могут производиться над ним.

Каждая функция $Function_{ij}$ описывается следующим образом: $Function_{ij}$ = <$Function\_Name_{ij}$, $Function\_Parameters_{ij}$, $Function\_Return\_Type_{ij}$>. $Function\_Name_{ij}$ – имя функции, $Function\_Parameters_{ij}$ – параметры функции, $Function\_Return\_Type_{ij}$ – тип значения, возвращаемого функцией, $Function\_Return\_Type_{ij} \in$ Func_Type.

Параметры функции $Function\_Parameters_{ij}$ могут состоять из множества её параметров, $Function\_Parameters_{ij} = \{Function\_Parameter_{ijk}\}_{k=0}^{funcparametercount}$. Каждый параметр функции $Function\_Parameter_{ijk}$ описывается своим типом и значением, т.е. $Function\_Parameter_{ijk}$= <$FuncParam\_Value_{ijk}$, $FuncParam\_Type_{ijk}$>, где $FuncParam\_Value_{ijk}$ – значение параметра, $FuncParam\_Type_{ijk}$ - тип параметра,

$FuncParam\_Type_{ijk} \in Func\_Type$. Тип параметра функции Func_Type может быть одним из следующих: $Func\_Type = String \cup Integer \cup Float \cup Boolean$.

В соответствии с метаонтологией разработана модель онтологии WIMP-интерфейсов, содержащая 51 элемент, подробно описанная в диссертационной работе. В автореферате приведёно описание элемента интерфейса «Дерево».

Дерево - элемент интерфейса, части которого иерархически организованы в качестве элементов дерева.

$Дерево = <UIElementType_{дерево}, Parameters_{дерево}, Events_{дерево}, Functions_{дерево}>$.

$UIElementType_{дерево}$ = Элемент управления.

$Parameters_{дерево} = Parameters_{составной\ элемент} \cup$ { Полосы прокрутки, Линии от корня, Линии между элементами, Подсветка элементов, Кнопки свёртки } $\cup$ Элементы дерева.

Полосы прокрутки = < Прокрутка, Boolean >. Описывает наличие полос прокрутки.

Линии от корня = < От корня, Boolean >. Описывает наличие линий, идущих от корня к элементам первого уровня иерархии.

Линии между элементами = < Линии между, Boolean >. Описывает наличие линий, идущих от элементов-родителей к элементам-потомкам.

Подсветка элементов = < Подсветка, Boolean >. Описывает, подсвечиваются ли элементы при наведении на них курсора мыши.

Кнопки свёртки = < Свёртка, Boolean >. Описывает наличие у каждого элемента дерева кнопок для переключения отображаемости его подэлементов.

Параметр «Элементы»: либо Элементы дерева = $\{<$ Элемент дерева$_i$, { Элемент дерева } $>\}_{i=0}^{elemcount}$, либо Элементы дерева = $\{<$ Элемент дерева$_i$, { Флажковый элемент дерева } $>\}_{i=0}^{elemcount}$. Описывает множество элементов дерева.

$Events_{дерево} = Events_{составной\ элемент}$.

$Functions_{дерево} = \varnothing$.

Разработанная модель онтологии не зависит от платформы и языка реализации интерфейса, является расширяемой (могут добавляться новые элементы интерфейса), так как описание элементов интерфейса производится в соответствии с метаонтологией, которая описывает их структуру.

В третьей главе описывается метод формирования модели WIMP-интерфейса в терминах моделей онтологий и метод генерации исходного кода WIMP-интерфейса по его модели.

Метод формирования модели WIMP-интерфейса в терминах моделей онтологий. Согласно онтологическому подходу к разработке пользовательского интерфейса, его

модель является конкретизацией моделей онтологий и состоит из следующих компонентов:

- *модели системы понятий диалога* (описывает множество терминов системы понятий диалога, необходимых для представления входных и выходных данных прикладной программы, обеспечения интеллектуальной поддержки действий пользователя в процессе его работы) и является конкретизацией модели онтологии системы понятий диалога;

- *модели WIMP-представления* (описывает структуру и свойства визуального представления элементов интерфейса; характеризуется множеством описаний окон $\text{WindowDescriptions}=\{<\text{UIElementName}_i, \text{Window}_i>\}_{i=1}^{windowcount}$) и является конкретизацией модели онтологии WIMP-интерфейсов;

- *модели связи интерфейса с прикладной программой* (описывает множество программных интерфейсов, предоставляемых прикладной программой, $\text{Interfaces}=\{\text{Interface}_i\}_{i=1}^{\text{int}\,erfacecount}$) и является конкретизацией модели онтологии связи интерфейса с прикладной программой;

- *модели сценария диалога* (описывает начальное окно StartWindow, множество состояний диалога States, а также действия, которые выполняются в каждом состоянии) и является конкретизацией модели онтологии сценария диалога.

Между компонентами модели WIMP-интерфейса существуют взаимосвязи. Так в модели WIMP-представления в качестве значений параметров элементов интерфейса могут использоваться имена элементов модели системы понятий диалога; в модели сценария диалога в качестве параметров инструкций могут использоваться элементы модели WIMP-представления и модели связи интерфейса с прикладной программой.

Структура модели WIMP-интерфейса, а также модель онтологии системы понятий диалога, связи интерфейса с прикладной программой и сценария диалога были разработаны и описаны в работах Грибовой В.В. Автором диссертационной работы предложен метод формирования каждого компонента модели WIMP-интерфейса по его структуре и моделям онтологий, который полностью описан в диссертационной работе. В автореферате приведено описание формирования каждого состояния $\text{State}_i \in \text{States}$ (фрагмент формирования модели сценария диалога). Для формирования каждого состояния State_i необходимо выполнить следующие действия:

1) Задать событие Event_i, $\text{Event}_i \in \bigcup_{\text{Window}_X \in \text{WindowDescriptions}} \text{Events}$

   (событие Event_i, принадлежит некоторому элементу интерфейса, определенному в модели WIMP-представления).

2) Задать множество переменных $Variables_i$, которые необходимы для описания последовательности инструкций $Instructions_i$, при этом для каждой переменной $Variable_{ij} \in Variables_i$ определить:

2.1) Имя переменной $Variable\_Name_{ij}$ - строковое значение.

2.2) Тип значения $Variable\_Type_{ij}$, т.е. $Variable\_Type_{ij} \in Var\_Type\_Mod$, где $Var\_Type\_Mod \subseteq Var\_Type$, $Var\_Type\_Mod = String \cup Integer \cup Float \cup Boolean \cup Enumerations \cup Windows$, где

$$Enumerations = \bigcup_{Window_X \in WindowDescriptions} Enumeration$$

(типом значения Enumerations являются такие перечислимые множества, которые определены в модели WIMP-представления),

$$Windows = \bigcup_{Window_X \in WindowDescriptions} Window_X$$

(типом значения Windows являются такие окна, которые определены в модели WIMP-представления).

3) Задать последовательность инструкций $Instructions_i$. Для формирования каждой инструкции $Instruction_{ij} \in Instructions_i$ выполнить Задание инструкции ($Instruction_{ij}$).

Метод генерации исходного кода WIMP-интерфейса по его модели. Согласно онтологическому подходу пользовательский интерфейс и прикладная программа разрабатываются раздельно (также как и в моделеориентированном подходе). Соответственно, исходный код программного средства состоит из: исходного кода прикладной программы, исходного кода WIMP-интерфейса, исходного кода связи интерфейса с прикладной программой.

Исходный код прикладной программы описывается на языке программирования её разработчиком. Исходный код WIMP-интерфейса состоит из: исходного кода структуры окон, исходного кода поведения окон, исходного кода инициализации интерфейса. Исходный код связи интерфейса с прикладной программой описывает программные интерфейсы, которая она предоставляет. Исходный код WIMP-интерфейса и исходный код связи интерфейса с прикладной программой генерируются автоматически по модели WIMP-интерфейса.

Одним из основных требований к системам автоматизации проектирования пользовательских интерфейсов является возможность генерации исходного кода на различные платформы и языки программирования. Однако методы программирования отличаются в различных языках программирования и платформах, что делает невозможным разработку генератора кода, способного генерировать исходный код на различные платформы и языки программирования, при этом

разработка множества генераторов кода для каждой пары «платформа - язык программирования» является крайне трудоёмкой. При этом для всех платформ и языков программирования можно выделить общие составные части исходного кода интерфейса, которые выполняют идентичные функции, но реализуются с помощью различных операторов и функций.

Решением данной проблемы является генерация кода не на язык программирования, а на специализированный абстрактный язык с последующим отображением кода на абстрактном языке в код на конкретный язык программирования.

Разработанный автором абстрактный язык описывает структуру исходного кода интерфейса и связи между его частями, не зависит от платформы и языка реализации интерфейса. Полная спецификация абстрактного языка приведена в диссертационной работе.

Код формируется на абстрактном языке, и состоит из следующих частей: кода инициализации интерфейса, кода программных интерфейсов, кода структуры окон и кода поведения окон. Процесс генерации каждой из частей полностью описан в диссертационной работе, при этом каждая часть кода может формироваться как на основе одного, так и на основе нескольких компонентов модели WIMP-интерфейса. Ниже приведён фрагмент, описывающий генерацию кода инициализации интерфейса.

main { **{ <**инициализация программного интерфейса**> }** run **<**имя**>**; }

1. $\{Interface_i \mid Interface_i = < Interfacename_i, InteractionModel_i, Functions_i>\}_{i=1}^{\text{int } erfacecount} \rightarrow$ **{ <**инициализация программного интерфейса**>}**, где $Interface_i$ – программный интерфейс МСПП. В зависимости от параметра $InteractionModel_i$ отображение имеет вид:

   1.1. $\{ Interface_i \mid InteractionModel_i = \text{Локальная}\}_{i=1}^{\text{int } erfacecount} \rightarrow$

   $\left\{ \begin{array}{l} Interfacename_1.initlocal.(\text{путь к файлу}_1, \text{имя класса}_1); \\ \ldots \\ Interfacename_{interfacecount}.\ initlocal.\ (\text{путь к файлу}_{interfacecount}, \text{имя класса}_{interfacecount}); \end{array} \right.$, где

   «путь к файлу»$_i \in$ IModelparameters, «имя класса»$_i \in$ IModelparameters модели взаимодействия $InteractionModel_i$;

   1.2. $\{ Interface_i \mid InteractionModel_i = \text{Распределенная} \}_{i=1}^{remotecount} \rightarrow$

   $\left\{ \begin{array}{l} Interfacename_1.initremote.(\text{сетевой адрес}_1, \text{номер порта}_1, \text{имя объекта}_1); \\ \ldots \\ Interfacename_{interfacecount}.\ initremote.(\ \text{сетевой адрес}_{interfacecount}, \text{номер порта}_{interfacecount}, \text{имя объекта}_{interfacecount}); \end{array} \right.$, где

«сетевой адрес»$_i$ ∈ IModelparameters, «номер порта»$_i$∈ IModelparameters, «имя объекта»$_i$ ∈ IModelparameters модели взаимодействия InteractionModel$_i$;

2. StartWindow → run StartWindowID; где StartWindow – это начальное окно МСД, StartWindowID - это идентификатор StartWindow.

Результатом процесса генерации кода является код на абстрактном языке, который не зависит от платформы и языка реализации интерфейса.

Четвёртая глава содержит описание требований к инструментальному средству Onto Dev для разработки, автоматической генерации и сопровождения WIMP-интерфейсов на основе онтологий, его архитектуры и методов реализации. На рис. 1 изображена архитектурно-контекстная диаграмма Onto Dev.

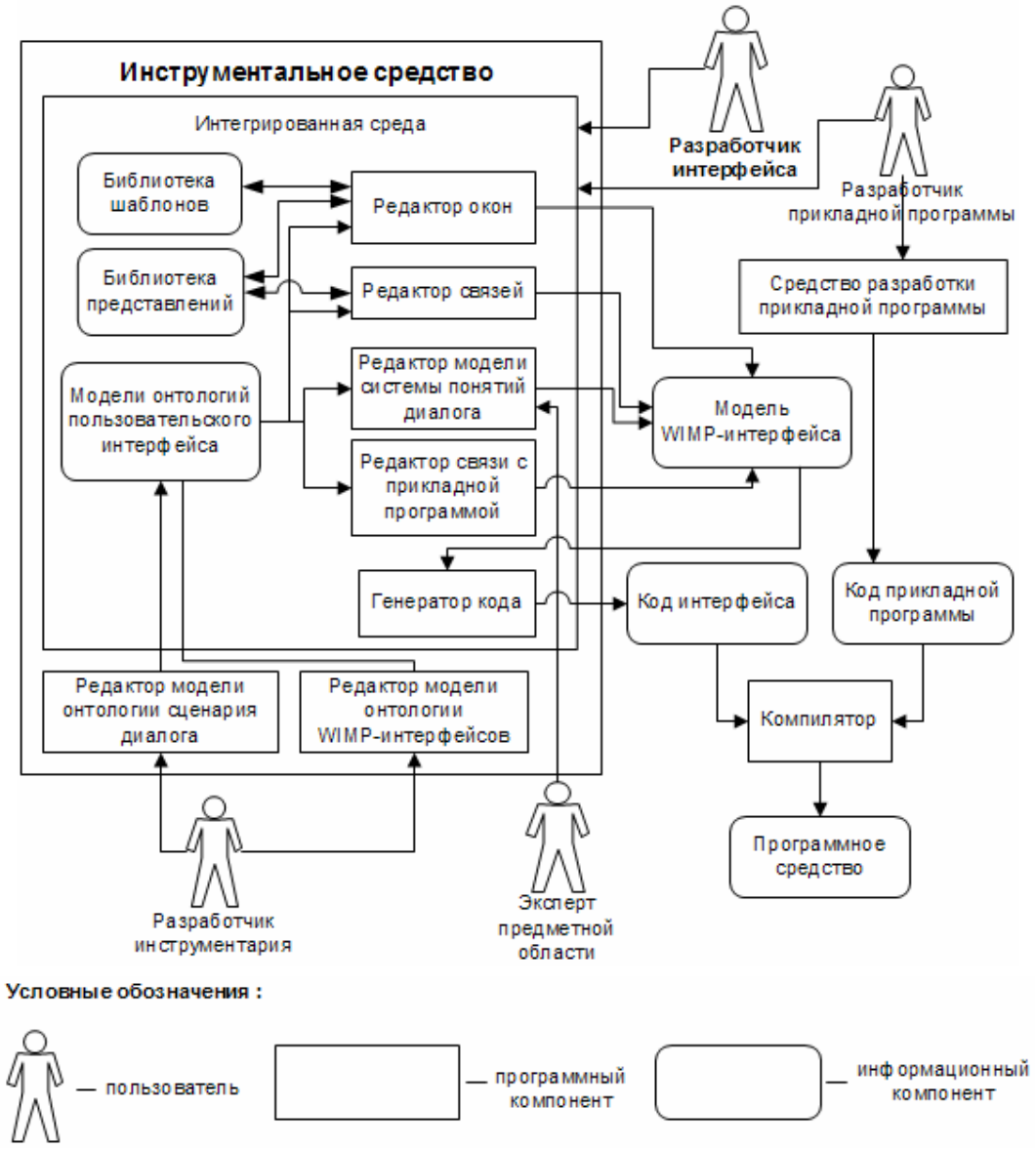

Рис. 1. Архитектурно-контекстная диаграмма Onto Dev.

Все информационные и программные компоненты, методы реализации каждого программного компонента, а также типы пользователей описаны в диссертационной работе.

Разработанное инструментальное средство Onto Dev удовлетворяет всем представленным требованиям: Onto Dev автоматически генерирует исходный код WIMP-интерфейса по его модели; обеспечивает генерацию WIMP-интерфейсов на различные языки и платформы: язык C# для платформы .NET 2 (включая .NET 2 Compact Framework – версию .NET для карманных компьютеров) и язык Java для платформы Java 2; поддерживается как локальное, так и сетевое взаимодействие интерфейса с прикладной программой по протоколам: TCP/IP и SOAP; все компоненты модели WIMP-интерфейса формируются с помощью структурных и визуальных редакторов, управляемых моделями онтологий; предоставляется возможность автоматического сопровождения модели WIMP-представления при изменении терминов модели системы понятий диалога или их структуры (автоматически изменяется модель WIMP-представления, при этом не требуется повторная генерация кода); предоставляются механизмы повторного использования фрагментов модели WIMP-представления, состоящих как из отдельных элементов интерфейса, так и их групп; инструментарий может расширяться добавлением новых элементов интерфейса, новых стандартных функций диалога, новых платформ и языков программирования.

В **пятой главе** описана технология проектирования и сопровождения WIMP-интерфейсов с помощью разработанного инструментального средства, описаны WIMP-интерфейсы, разработанные с его помощью, приведена сравнительная оценка трудоёмкости разработки описанных интерфейсов с помощью традиционных средств и с помощью предложенного в работе инструментального средства.

Предлагаемая в диссертационной работе технология разработки WIMP-интерфейсов основана на принципе независимого проектирования интерфейса и прикладной программы с последующим их связыванием. Этапы разработки WIMP-интерфейса с помощью Onto Dev подробно описаны в диссертационной работе, на рис. 2 изображена общая схема разработки.

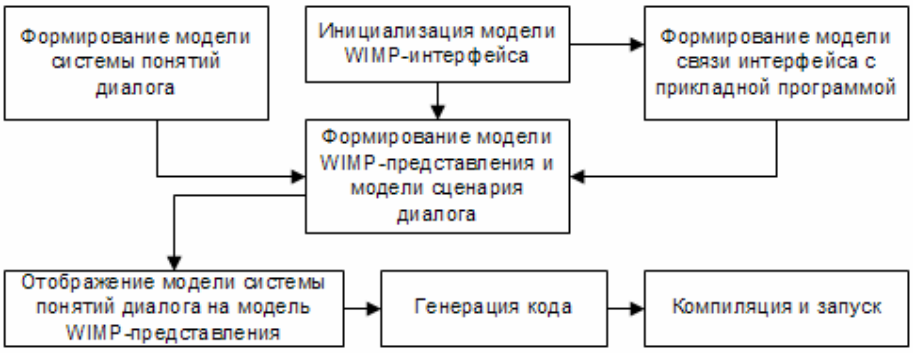

Рис. 2. Общая схема разработки WIMP-интерфейса с помощью Onto Dev.

Инструментарий разработчика интерфейса может расширяться следующими способами: расширением библиотеки шаблонов, расширением библиотеки представлений, расширением модели онтологии WIMP-интерфейсов, расширением модели онтологии сценария диалога, расширением множества доступных языков программирования и платформ в генераторе кода. Все способы расширения инструментария разработчика описаны в диссертационной работе.

Инструментарий для разработки пользовательского интерфейса, представленный в данной работе, использовался при разработке WIMP-интерфейса системы интеллектуальной поддержки обследования больных для врача-уролога. Назначение системы – обеспечить интеллектуальную поддержку врачу-урологу при формировании и ведении историй болезни, а также организовать компьютерный архив историй болезни, предназначенный для статистической обработки и создания отчетов. История болезни формируется на основе модели системы понятий диалога (базы наблюдений) в области урологии. База наблюдений содержит более 700 терминов и около 5000 вариантов значений. Система представлена в двух версиях: локальной и распределённой. Система интеллектуальной поддержки используется в урологическом отделении Городской клинической больницы N2 г. Владивостока.

Также инструментарий использовался для разработки WIMP-интерфейса для системы анализа газетных объявлений о купле-продаже недвижимости. Предметной областью здесь является торговля недвижимостью в пределах отдельно взятого города. Назначением системы является: анализ текстов газет и выборка объявлений, содержащих информацию о купле-продаже недвижимости; определение приблизительного местоположения недвижимости на основе адресного плана города; группировка объявлений по их местоположению; генерация отчётов и их распечатка. В ходе разработки был сформирован адресный план города Владивостока, который содержит информацию о приблизительно 700 улицах города, 40 микрорайонах города, при этом для каждой улицы выделено до 5 различных вариантов сокращения её названия. Система анализа газетных объявлений о купле-продаже недвижимости используется в ООО «Барс».

ОСНОВНЫЕ РЕЗУЛЬТАТЫ РАБОТЫ

1. Разработана модель онтологии WIMP-интерфейсов, состоящая из двух уровней: метаонтологии, предназначенной для описания структуры элементов интерфейса и непосредственно онтологии, содержащей описание множества элементов интерфейса. Модель онтологии не зависит от платформы и языка реализации интерфейса.

2. Разработан метод формирования модели WIMP-интерфейса в терминах моделей онтологий.
3. Разработан метод генерации исходного кода WIMP-интерфейса по его модели. Генерация осуществляется на абстрактный язык, который не зависит от платформы и языка реализации интерфейса, с последующим отображением в код на конкретные языки программирования.
4. Разработаны методы реализации инструментального средства Onto Dev, предназначенного для проектирования, автоматической генерации и сопровождения WIMP-интерфейсов на основе онтологий. Onto Dev обеспечивает: автоматическую генерацию исходного кода по модели WIMP-интерфейса на языки C# для платформы .NET 2 (включая .NET 2 Compact Framework) и язык Java для платформы Java 2; организацию как локального, так и сетевого взаимодействия интерфейса с прикладной программой; повторное использование компонентов модели WIMP-интерфейса; расширение самого инструментального средства Onto Dev.
5. Предложена технология проектирования и сопровождения WIMP-интерфейсов с помощью Onto Dev. Проектирование и сопровождение осуществляется с помощью визуальных и структурных редакторов в терминах систем понятий специалистов (экспертов предметной области, дизайнеров, программистов). Проведена практическая проверка Onto Dev: разработаны WIMP-интерфейсы для Системы интеллектуальной поддержки обследования больных для врача уролога и Системы анализа газетных объявлений о купле-продаже недвижимости.

ОСНОВНЫЕ ПУБЛИКАЦИИ ПО ТЕМЕ ДИССЕРТАЦИИ

1. Тарасов А.В. Модель выразительных средств интерфейса при его разработке на основе онтологий // Информатика и системы управления. – 2005. №1. – с. 97-106.
2. Грибова В.В., Тарасов А.В. Модель управления компонентом представления информации в пользовательском интерфейсе на основе онтологии: Труды конференции. Первая международная конференция "Системный анализ и информационные технологии" - 2005 - М.: КомКнига, 2005. с. 184-187.
3. Грибова В.В., Тарасов А.В. Генератор пользовательского интерфейса, управляемый онтологиями: Материалы международной научной конференции. Интеллектуальные и многопроцессорные системы - 2005 – Таганрог: Изд-во ТРТУ, 2005. – с. 316-321.
4. Грибова В.В., Тарасов А.В. Модель онтологии предметной области «Графический пользовательский интерфейс» // Информатика и системы управления. – 2005. №1. - с. 80–90.

5. Грибова В.В., Тарасов А.В. Генератор кода пользовательского интерфейса, управляемый онтологией // Искусственный интеллект. – 2005. №4. - с. 457-464.
6. Грибова В.В., Тарасов А.В. Концепция расширяемого инструментария для разработки пользовательского интерфейса: Труды конференции. Научная сессия МИФИ-2006. – М.: МИФИ, 2006. - с. 132-133.
7. Тарасов А.В. Среда разработки пользовательских интерфейсов OntoDev: Сборник докладов. Открытый дальневосточный конкурс студентов, аспирантов и молодых специалистов "Программист-2006". - Владивосток: ИАПУ ДВО РАН, 2006. – с. 84-88.
8. Грибова В.В., Тарасов А.В. Управление процессом автоматической генерации программного кода пользовательского интерфейса по его модели [Электронный ресурс]: III Международная конференция «Параллельные вычисления и задачи управления».– 2006 – (CD-ROM).
9. Грибова В.В., Тарасов А.В. Инструментальное средство ONTODEV для проектирования и автоматической генерации пользовательского интерфейса // Информатика и системы управления. - 2006. №1. - с. 152-158.
10. Грибова В. В., Тарасов А. В. Гибкие инструментальные средства для разработки пользовательского интерфейса // Приборостроение. - 2007. №3. - с. 35—38.
11. 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.

Личный вклад автора. Все результаты, составляющие основное содержание диссертации, получены автором самостоятельно. В работах [2-6, 8] автором описана модель онтологии WIMP-интерфейсов, определены метод формирования модели WIMP-представления и метод генерации исходного кода WIMP-интерфейса по его модели. В работах [9-10] автором описаны методы реализации инструментального средства Onto Dev. В работе [11] автором разработаны методы реализации WIMP-интерфейса Системы интеллектуальной поддержки обследования больных и методы её реализации.

ТАРАСОВ Алексей Владимирович

РАЗРАБОТКА И ИССЛЕДОВАНИЕ МЕТОДОВ ГЕНЕРАЦИИ И СОПРОВОЖДЕНИЯ WIMP-ИНТЕРФЕЙСОВ

Автореферат
диссертации на соискание ученой степени
кандидата технических наук

Подписано к печати 07.09.2007 г.	Усл.п.л. 1,0	Уч.-изд.л.0,75
Формат 60x84/16	Тираж 100.	Заказ .

Издано ИАПУ ДВО РАН. Владивосток, Радио, 5
Отпечатано участком оперативной печати ИАПУ ДВО РАН, Владивосток, Радио, 5