OSTIS-2011
(Open Semantic Technologies for Intelligent Systems)

УДК 004.5, 004.8

АВТОМАТИЗАЦИЯ РАЗРАБОТКИ ПОЛЬЗОВАТЕЛЬСКИХ ИНТЕРФЕЙСОВ С ДИНАМИЧЕСКИМИ ДАННЫМИ

В.В. Грибова (*gribova@iacp.dvo.ru*)
Н.Н. Черкезишвили (*nickolayc@gmail.com)*

Учреждение Российской академии наук Институт автоматики и процессов управления Дальневосточного отделения РАН, г. Владивосток, Россия

В данной работе предлагается концепция автоматизации разработки пользовательских интерфейсов с динамическими данными, допускающих формирование наборов входных/выходных данных динамически во время работы приложения. Современные средства поддерживают автоматическую генерацию только интерфейсов со статическими данными. Предлагаемая в работе концепция направлена на устранение их недостатков.

Ключевые слова: онтологический подход, пользовательский интерфейс, автоматическая генерация.

Введение

Разработка пользовательских интерфейсов является очень трудоемкой задачей и по оценкам специалистов для сложных и комплексных программных систем трудозатраты на разработку пользовательского интерфейса занимают до 70% общего времени разработки. Сопровождение пользовательских интерфейсов также является сложной задачей, поскольку требования пользователей, среда использования программного средства, расширение его функциональности требуют модификации пользовательского интерфейса. Для снижения трудоемкости их разработки и сопровождения в настоящее время существуют средства автоматизации проектирования и реализации: построители WIMP-интерфейсов, моделеориентированные средства и средства, основанные на онтологическом подходе [Грибова и др., 2001].

Однако все перечисленные средства ориентированы на автоматизацию разработки пользовательских интерфейсов со статическими данными, для которых сценарий диалога и визуальное представление полностью определяется на этапе проектирования интерфейса. В то же время для редакторов, программ, в которых наборы входных/выходных данных генерируются логикой приложения, а также гибко конфигурируемых приложений наборы входных/выходных данных, структуру каждого набора, а также сценарий диалога невозможно определить на этапе проектирования интерфейса. Такие интерфейсы называют интерфейсами с динамическими данными.

Реализация интерфейсов с динамическими данными либо полностью осуществляется на языках программирования (C++, Java, Pascal и др.), либо некоторые компоненты интерфейса реализуются с использованием специализированных средств автоматизации разработки, остальные реализуются на языках программирования. В результате проектирование, реализация и особенно сопровождение пользовательских интерфейсов оказывается чрезвычайно трудоемкими.

Целью данной работы является представление концепции автоматизации разработки интерфейсов с динамическими данными.

1. Онтологический подход для автоматической генерации пользовательских интерфейсов

Предлагаемая в работе концепция автоматизации разработки интерфейсов с динамическими данными является развитием онтологического похода [Грибова и др., 2005, Грибова и др., 2006]. Инструментарий, основанный на данном подходе, широко используется для автоматической генерации пользовательских интерфейсов; с его помощью разработаны пользовательские интерфейсы для приложений в различных предметных областях.

Основными положениями автоматизации проектирования, реализации и сопровождения пользовательского интерфейса на основе онтологического подхода являются:

- проведение анализа профессиональной деятельности, связанной с разработкой пользовательского интерфейса: выделение групп специалистов, осуществляющих разработку и сопровождение пользовательского интерфейса, а также систем понятий, которые они используют в своей работе;
- построение онтологий пользовательского интерфейса, необходимых для того, чтобы в их терминах разработчики интерфейса могли определять и модифицировать структуру конкретной модели пользовательского интерфейса;
- разработка модели пользовательского интерфейса в терминах онтологий, которая является конкретизацией онтологий пользовательского интерфейса;
- построение алгоритма автоматического преобразования модели интерфейса в программный код, который управляется онтологиями пользовательского интерфейса, при этом характеристики конкретной модели являются входными данными для этого алгоритма.

Таким образом, в онтологическом подходе модель интерфейса является основной информационной составляющей, на основе которой автоматически генерируется код интерфейса. Модель интерфейса состоит из нескольких компонентов, описывающих всю необходимую информацию о пользовательском интерфейсе, необходимую для его генерации. (Рис. 1).

Компоненты модели интерфейса создаются разработчиками интерфейса (экспертами предметной области, дизайнерами, программистами) с помощью редакторов, управляемых соответствующими онтологиями (модель задач формируется с помощью редактора, управляемого онтологией задач, модель системы понятий диалога формируется с помощью редактора, управляемого онтологией системы понятий диалога и т. д.). В результате разработчики интерфейса освобождаются от изучения специализированных языков спецификаций и программирования и в терминах понятных им систем понятий формируют соответствующие компоненты модели интерфейса. После формирования модели интерфейса генератор кода пользовательского интерфейса автоматически генерирует код интерфейса, в который интегрируется код логики приложения, затем полученный код компилируется в приложение, с которым работает пользователь.

2. Основные положения концепции

Концепция разработки интерфейсов с динамическими данными расширяет модель интерфейса онтологического подхода новыми компонентами и, соответственно, онтологиями, в терминах которых новые компоненты будут формироваться и модифицироваться [Грибова и др., 2009]. Для описания модели интерфейса с динамическими данными были выделены статические характеристики модели интерфейса, которые не изменяются в процессе работы приложения, и динамические характеристики модели интерфейса, которые определяют структуру возможных изменений, но конкретные их значения определяются в процессе работы приложения.

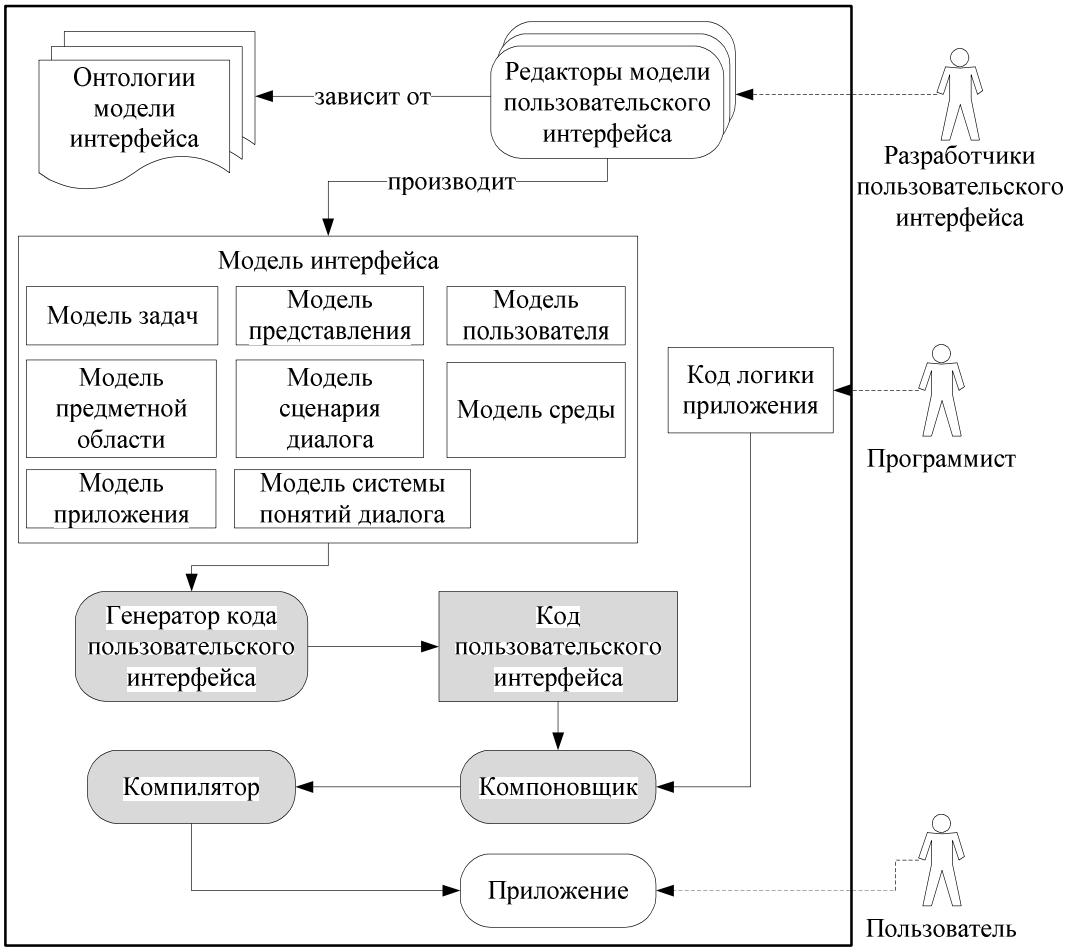

Рисунок 1 - Концептуальная схема онтологического подхода

Анализ динамических характеристик интерфейса привёл к следующим изменениям:

I. Онтология представления пользовательского интерфейса заменена онтологиями абстрактного и конкретного представления. Онтология представления онтологического подхода предназначена для формирования модели представления в терминах WIMP-интерфейсов (кнопки, поля ввода, списки, чекбоксы и т.д.) [Гультяев и др., 2000, Мандел, 2001]. Однако, в силу специфики приложений с динамическими данными, для таких приложений невозможно определить их конкретное представление во время проектирования.

Онтология абстрактного представления позволяет описывать представление интерфейса в терминах абстрактного пользовательского интерфейса [Hallvard Trætteberg и др., 2002], независимого от среды исполнения приложения и типов входных/выходных данных интерфейса. Элементы абстрактного интерфейса позволяют определять структуру представления информации, не уточняя их конкретное представление.

Онтология конкретного представления предназначена для формирования конечного представления пользовательского интерфейса, описанного в терминах WIMP-интерфейсов. Модель конкретного представления автоматически формируется во время работы приложения по модели абстрактного представления, соответствиям между: онтологиями пользователя и конкретного представления, онтологиями абстрактного и конкретного представлений, а также по конкретным характеристикам входных/выходных данных, например, количества элементов ввода (влияет на выбор элемента представления).

II. Онтологии задач пользователя (статическая) и сценарий диалога заменены онтологией динамических задач. Онтология задач пользователя (статическая) онтологического подхода предназначена для описания иерархии задач пользователя и связей между задачами, определяющих порядок и условия их выполнения: объединение (обе задачи в связке могут

выполняться параллельно), выбор (в один момент времени может выполняться только одна задача связки), допуск (устанавливает строгую последовательность выполнения задач), деактивация (выполнение одной задачи делает другую задачу недоступной для выполнения).

Онтология сценария диалога напрямую связана с онтологией представления и предназначена для описания множество возможных состояний диалога и переходов из состояния в состояние. Состояния диалога и условия переходов описываются в терминах интерфейсных элементов модели представления.

В динамических приложениях на этапе проектирования невозможно задать интерфейсные элементы, и, соответственно, сценарий диалога. Введение в онтологию задач динамических характеристик и правил выполнения задач позволяет задавать сценарии выполнения задач, заместив тем самым модель сценария диалога.

К динамическим характеристикам относятся:

- переменные задачи; переменные определяют динамический контекст задачи, формируемый в процессе её выполнения. Переменные могут хранить промежуточные данные выполнения алгоритмов и влиять на сценарии выполнения задач. Выделены следующие типы переменных: логические, целочисленные, строковые, действительные и перечислимые. Перечислимые переменные определяются разработчиками модели задач и могут содержать множество допустимых значений для каждого типа перечисления, например, перечисление «дни недели» содержит множество значений {понедельник, вторник, среда, …}. Перечислимые переменные могут принимать одно из множества значений, относящихся к этому типу.

- атрибуты задач; для каждой задачи определены два типа атрибутов: «состояние» и «статус задачи». Атрибут «состояние» может принимать одно из следующих значений: задача выполнена, задача не выполнена и задача прервана. Атрибут «статус» может принимать одно из следующих значений: задача доступна, задача недоступна и задача в процессе выполнения.

- правила выполнения задач представляют собой пару вида: <условие, действие>. Условие является логическим выражением вида «если (условие1) И | ИЛИ … (условиеN)». Действие представляет собой набор инструкций, которые выполняются только в том случае, если условие истинно. В условиях правил выполнения задач указываются значения атрибутов задач и переменных, в зависимости от значений которых выполняются действия: изменение значений атрибутов задач и переменных, вызовы системных функций интерфейса [Грибова и др., 2007] либо функций логики приложения.

Онтология модель задач (статическая) также расширена рядом дополнительных возможностей:

- добавлен новый тип задач – «задача приложения»; задачи приложения в отличие от задач пользователя не связаны с пользовательским интерфейсом, а связаны с логикой приложения и предназначены для автоматизации важных для интерфейса функций управления наборами входных/выходных данных задач (сохранение, редактирование, просмотр, удаление данных и др.);

- к атрибутам задач каждого типа добавлен атрибут «входные/выходные данные», необходимый для установки связи между элементами модели системы понятий диалога и логикой приложения.

III. Онтология пользователя расширена онтологией ролей пользователей. Онтология пользователя онтологического подхода позволяет описывать характеристики пользователя, такие как, уровень опытности, пол, возраст, предпочтение ввода, профессия пользователя и другие. Эти характеристики составляют портрет пользователя, необходимый для генерации пользовательского интерфейса.

Средства онтологического подхода позволяют разрабатывать только однопользовательские приложения, однако они не предназначены для генерации приложений, у которых разным типам пользователей доступна различная функциональность. В связи с этим, онтология пользователя была расширена онтологией ролей пользователей, позволяющей задавать для каждого пользователя/группы пользователей допустимую функциональность.

Таким образом, полученная в результате преобразований модель интерфейса (каждый компонент модели интерфейса является результатом конкретизации соответствующей онтологии) состоит из следующих компонентов:

- динамической модели задач, включающей модель задач пользователя, приложения и динамических характеристик задач;
- модели системы понятий диалога, описывающей множество понятий некоторой предметной области, связи между этими понятиями;
- модели среды, описывающей характеристики среды исполнения приложения;
- модели пользователя, состоящей из модели профиля пользователя и модели ролей;
- модели представления, состоящей из модели абстрактного представления и модели конкретного представления

IV. Генератор кода заменён интерпретатором модели интерфейса. При использовании генератора кода интерфейса, прежде чем пользователь получает исполнимое приложение, необходимо выполнить генерацию кода приложения, компоновку и компиляцию кода. Только после выполнения этих действий пользователь может начать работать с приложением. Таким образом, при каждом изменении модели интерфейса, необходимо повторять эти действия заново, что неизбежно влечёт за собой перезапуск приложения. Но в интерфейсах с динамическими данными, изменения в модели интерфейса могут происходить очень часто в процессе работы пользователя с приложением. Каждое такое изменение будет требовать от пользователя повторного выполнения описанной выше последовательности действий, необходимых, чтобы изменения в модели интерфейса вступили в силу.

Интерпретатор в отличие от генератора кода не генерирует код интерфейса (Рис. 2), а интерпретируют модель интерфейса сразу в исполняемое приложение. Это позволяет мгновенно отражать изменения в пользовательском интерфейсе при изменении модели интерфейса, не требуя перезапуска приложения.

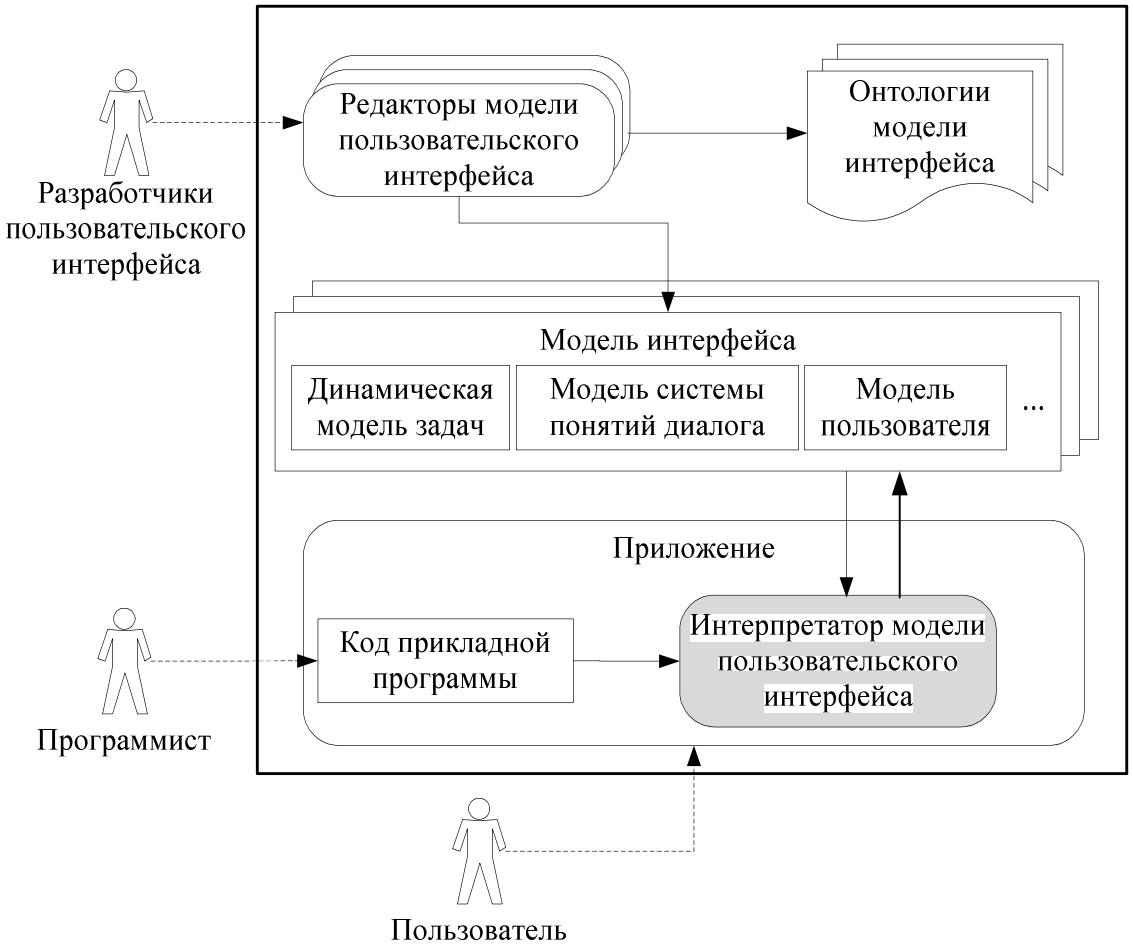

Рисунок 2 - Схема автоматической генерации интерфейсов с динамическими данными

Заключение

Разработка приложений с динамическими данными является очень трудоемкой задачей, при этом существующие средства автоматической генерации пользовательских интерфейсов не позволяют автоматизировать их разработку. Предлагаемая в работе концепция является расширением онтологического подхода к разработке и автоматической генерации пользовательских интерфейсов со статическими данными и предназначена для автоматизации разработки интерфейсов с динамическими данными. В настоящее разрабатывается инструментарий, основанный на предложенной концепции.

Благодарности

Работа выполнена при финансовой поддержке РФФИ, проект "Интеллектуальные многоагентные системы для управления распределенной обработкой онтологий, знаний и данных" (10-07-00090-а), и ДВО РАН по Программе №15 ОЭММПУ, проект "Модели мультиагентных систем для управления распределенной обработкой информации" (09-I-ОЭМПУ-02).

Библиографический список

[Грибова и др., 2001] Грибова В.В., Клещев А.С. Методы и средства разработки пользовательского интерфейса: современное состояние // Программные продукты и системы, 2001. №1. С. 2–6.

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

[Грибова и др., 2006] Грибова В.В., Клещев А.С. Управление проектированием и реализацией пользовательского интерфейса на основе онтологий // Проблемы управления №2. 2006. С.58-62

[Грибова и др., 2009] Грибова В.В., Черкезишвили Н.Н. Концепция автоматизации проектирования и реализации пользовательских интерфейсов с динамическими данными // Материалы рабочего семинара «Наукоёмкое программное обеспечение» в рамках 7-й международной конференции памяти академика А.П. Ершова «Перспективы систем информатики».– Новосибирск: ООО «Сибирское научное издательство», 2009.- С.97-103.

[Гультяев и др., 2000] Гультяев А.К., Машин В.А. Проектирование и дизайн пользовательского интерфейса.- СПб.: КОРОНА принт. 2000.- 352 с.

[Мандел, 2001] Мандел Т. Дизайн интерфейсов: Пер. с англ.- М.: ДМК Пресс, 2005.- 416 с.

[Hallvard Trætteberg и др., 2002] Hallvard Trætteberg. Model-based User Interface Design.- Information Systems Group Department of Computer and Information Sciences. 2002. 211 с.

[Грибова и др., 2007] Грибова В.В., Тарасов А.В. Гибкие инструментальные средства для разработки пользовательского интерфейса// Известия вузов. Приборостроение. 2007. Т. 50, № 3. С. 35—38.