БИБЛИОГРАФИЧЕСКИЙ СПИСОК

1. *Блушвили И.В.* Децентрализованный алгоритм организации сети распределенных вычислений и сравнительный анализ с передовыми алгоритмами.// Интеллектуальные и многопроцессорные системы. // Материалы международной научной конференции. Т.1. Таганрог: Изд-во ТРТУ, 2004. 496 с.
2. *Мельник Э.В., Пуха И.С.* Об одном подходе к организации распределенных баз данных.// Интеллектуальные и многопроцессорные системы. // Материалы международной научной конференции. Т.2. Таганрог: Изд-во ТРТУ, 2003. 309 с.
3. *Пуха И.С.* Мультиагентный подход в системах управления и хранения информации // // Российская Академия Наук, Южный Научный Центр. // Материалы второй ежегодной научной конференции. Ростов-на-Дону: Изд-во ЮНЦ РАН, 2006. 256 с.

В.А. Аграновский, В.И. Баранец, Р.Н. Селин

ОБОБЩЕННЫЙ МЕТОД ПОСТРОЕНИЯ ПРИКЛАДНЫХ И СЕТЕВЫХ ИНТЕРФЕЙСОВ ДЛЯ СУБД-ПРИЛОЖЕНИЙ

Разработка интерфейса – важнейшая составляющая процесса создания СУБД-приложения. Ключевыми требованиями при создании интерфейса СУБД-приложения являются, во-первых, возможность легкой и надежной модификации этого интерфейса, а во-вторых, независимость от инструментальной среды программирования. Для удовлетворения этих требований предлагается следующий подход: интерфейс приложения описывается формально в виде некоторой синтаксической структуры на языке XML [1].

Данный интерфейс является конкретным экземпляром интерфейса приложения (объектом) в рамках некоторой модели. Эта модель определяется так же формально как схема XML структур в виде XSD файла [2].

Формальная модель интерфейса приложения строится как некоторое дерево, узлы которого могут иметь атрибуты. Это дерево может иметь рекурсивную структуру. Поскольку задача автоматической генерации программы, реализующей интерфейс приложения, сводится к некоторому преобразованию одного атрибутированного дерева в другое атрибутированное дерево, для такого преобразования можно использовать существующий набор стандартных трансформаций над такими деревьями, а также мощный математический аппарат теории графов, предназначенный для работы с древовидными структурами.

Принципы обработки электронных документов

Принцип обработки XML-документов заключается в следующем: при разборе XML-документа программа-анализатор каждому элементу, найденному в XML-дереве должна поставить в соответствие набор элементов, определяющих форматирование этого элемента в синтаксисе конкретной среды программирования. Другими словами, задается шаблон форматирования для XML- элементов, причем сам этот шаблон может иметь структуру соответствующего фрагмента XML-документа.

Инструкции программы-анализатора должны определять точное месторасположение элемента XML в дереве, в результате чего появится возможность применять различные стили оформления к одинаковым элементам в зависимости от контекста их использования. На рис. 1 представлена схема, иллюстрирующая процесс обработки программой-анализатором MSXSL XML-документа на основе XSL-правил преобразования для конечной среды HTML.

Среди работ по созданию формального языка описания интерфейсов приложений наиболее удачна реализация языка UIML – «Аппаратно-независимый XML язык пользовательских интерфейсов» [3].

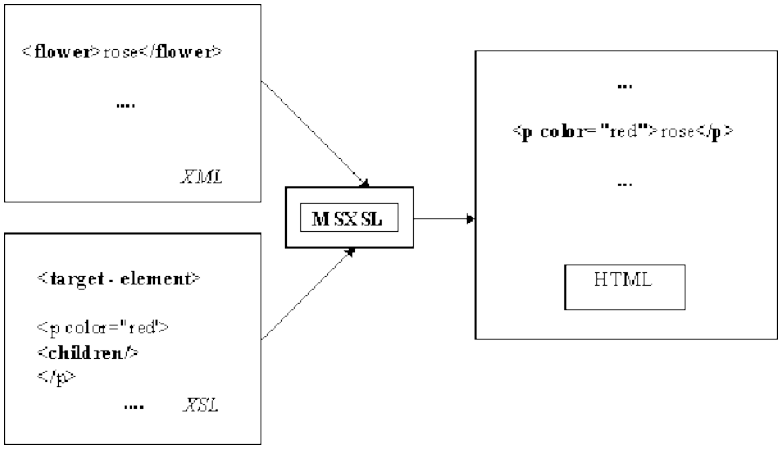

Рис. 1. Архитектура обработки электронных документов

Данный язык является языком высокого уровня, который не зависит от типа устройств, то есть в равной степени может применяться как на персональных компьютерах, так и на сотовых телефонах. Однако данный язык не совсем подходит именно для СУБД-приложений. Например, практически каждая экранная форма имеет «data environment» – некоторый контекст оперирования данными, который формируется из описания соединения с базой данных и некоторых данных.

Исходя из практических соображений сформулируем требования к языку описания интерфейсов СУБД-приложений:

– язык должен быть достаточно высокоуровневым, чтобы не зависеть от среды программирования. Это позволит упростить задачи изменения (в том числе и сопровождения) СУБД-приложения, с одной стороны, при обновлении версий программного обеспечения и, с другой стороны, между различными средами разработки (например при эволюционной смене технологий);

– должна обеспечиваться легкость и эффективность по скорости модификации готового кода. Это должно выполняться, чтобы уменьшить затраты на сопровождение СУБД-приложений по сравнению с традиционными технологиями программирования;

– необходимо разделение между описанием собственно интерфейса и логикой приложения. Это позволит также сократить трудозатарты на модификацию готового СУБД-приложения. Особенно такой подход позволит повысить эффективность работы на этапе внедрения программных комплексов.

Применение языка XML для описания интерфейса обеспечивает независимость от среды программирования. Реализация получаемого описания интерфейса может осуществляться двумя способами: статическим и динамическим.

В случае статической реализации интерфейса используется набор различных правил его преобразования в конкретную среду программирования. Возможный набор подобных трансформаций можно определить с помощью языка XSLT [4]. Уместно сравнить роль и назначение языков SQL, описанного, например, в [5], и XSLT.

Общеизвестно, что SQL стал стандартным декларативным высокоуровневым языком для решения любых задач в среде реляционной модели данных. Язык XSLT

является стандартным декларативным высокоуровневым языком для решения любых задач в среде иерархической модели данных. Подобно тому как существует множество систем управления базами данных, в основе которых лежит язык SQL, уже существует и появляется все больше реализаций XSLT процессоров.

Таким образом, построение СУБД-приложения осуществляется для различных сред по одинаковой схеме (см. рис. 2):

- создание XML-описания интерфейса;
- трансформация описания в конкретную среду программирования;
- модификация кода в соответствии с логикой приложения;
- итоговая генерация приложения.

При данном подходе легкость модификации готового исходного кода обеспечивается автоматической процедурой трансформации описания.

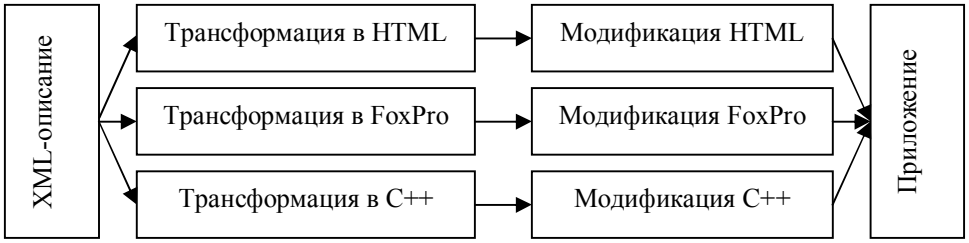

Рис. 2. Трансформация документов-прототипов в интерфейс

Достоинствами подхода являются, во-первых, сокращение время разработки СУБД-приложения за счет автоматизации процедуры создания идентичных наборов операторов для идентичных экранных компонентов, а во-вторых, автоматическая процедура трансформации устраняет возможность внесения оператором ошибок (например, синтаксических) на этапе кодирования.

Применение различных процедур трансформации позволяет получать по одной спецификации приложения для различных сред разработки. Недостатком же такого подхода является то, что необходимо предусмотреть специальный механизм, который будет отслеживать, чтобы при внесении изменений в описание не терялись и оставались корректными проделанные модификации.

Другой подход основан на динамической генерации приложения. В его рамках также создается XML-описание интерфейса. Дальнейший процесс построения приложения отличается от предыдущего сценария. Существует некоторый программный комплекс, осуществляющий синтаксический разбор входных структур, описывающих интерфейс СУБД-приложения, сборку приложения по этому описанию на этапе запуска/работы и функциональность готового СУБД-приложения.

Рассмотрим подробнее структуру модели интерфейсов. В логическом представлении комплекса можно выделить три логически независимые части:

- структура интерфейса;
- внешний вид (стиль) компонентов;
- функциональное поведение компонентов.

Эти части реализуются в виде подключаемых компонентов (в западной литературе именуемых плагин-модулями). Данный подход позволяет изменять внешний вид и функциональность СУБД-приложения не только от запуска к запуску, но и в момент работы приложения (например, внешний вид формы может измениться уже при ее повторном показе). Функциональность СУБД-приложения осуществляется в событийно-ориентированном стиле.

События являются единственным средством вызова процедур. Каждое событие может иметь любое количество обработчиков. Каждое событие имеет стандартный обработчик (возможно, ничего не делающий – своего рода “пустышку”). При генерации события объект обращается к диспетчеру событий, передавая ему некоторый набор параметров. Их описание и количество определяется общими требованиями к комплексу и функциональности объекта.

Диспетчер событий на основе этих параметров определяет дальнейшие действия СУБД-приложения: выполняются стандартный обработчик события и, при наличии, пользовательские обработчики (подключенные дополнительные модули). Причем порядок выполнения регулируется задаваемыми программистом-разработчиком приоритетами.

Достоинством динамического подхода по сравнению со статическим является отсутствие необходимости специализированного механизма отслеживания пользовательских изменений кода. Следует отметить, что использование идеологии подключаемых модулей в совокупности с централизованным автономным диспетчером событий открывает возможности для реализации данного подхода не только в интерпретирующих, но и в компилирующих средах.

Следует отметить, что рассматриваемые подходы автоматической генерации обладают также некоторыми недостатками. В совокупности с легкостью и быстротой модификации программного обеспечения мы получаем открытость части исходных текстов, что может быть весьма нежелательным фактором при распространении и эксплуатации комплекса. Для решения данной проблемы могут использоваться различные методы [6, 7], но ее детальное обсуждение выходит за рамки данной статьи.

Попробуем рассмотреть подробнее саму структуру интерфейса. Предлагаемый интерфейс формируется из нескольких частей. Первая часть «description» – описательная, в нее входит такая информация как название приложения «appname», его версия «version», авторы «authors» и, опционально, дополнительные сведения «memo».

Отличительная особенность СУБД-приложения от приложений других видов – наличие так называемого «data environment», некоторого контекста оперирования данными. Данный контекст формируется из некоторых параметров, например, описания соединения с базой данных. Каждый параметр «parameter» контекста данных уровня приложения «appdataenv» состоит из названия параметра «name» и его значения «value».

Data environment может быть двух видов – уровня приложения и уровня экранной формы. Основное отличие состоит в том, что на первом уровне описываются данные, общие для всего приложения, тогда как второй уровень включает в себя параметры, характерные только для конкретной экранной формы.

Структура экранной формы состоит из трех частей. Первая часть содержит описание «screenform data environment» – контекста данных экранной формы «sfdataenv». В ней используются расширенные свойства «application data environment» – контекста данных всего СУБД-приложения, который включает в себя параметры для формирования подключения к базе данных. По своей структуре «sfdataenv» аналогична «appdataenv».

Вторая часть – это декларативная непроцедурная спецификация структуры экранной формы. Эта часть – «structure» – не зависит от используемой инструментальной системы программирования и определяется в содержательных терминах исходя из потребностей задачи. Реализация этой части представляет собой рекурсивное дерево экранных компонентов «control», представляющих собой стандартные визуальные компоненты, практически одинаковые для большинства популярных сред разработки. Экранный компонент обязательно должен иметь уникальное в рамках реализации приложения название «name» и тип «type», представляющий собой один из стандартных типов визуальных компонентов, таких как Label, TextField, ComboBox и т.п.

Другой подход для представления структуры – различные экранные компоненты представлять не в виде универсального «control» с индивидуальными наборами свойств для каждого компонента, а для любой экранный компонент имеет свое представление в контексте описания интерфейса СУБД-приложения.

Третья часть экранной формы – «style» – содержит стилевую часть описания интерфейса. В данной части содержится также инструментально-независимая спецификация, описывающая внешний вид экранной формы. Здесь описываются такие характеристики экранных компонентов как размеры, цвет, взаимное расположение и т.п. Формально данная часть представляет собой некоторый набор свойств, привязывающихся к конкретным элементам экранных форм. Посредством применения данного стилевого описания внешний вид приложения может кардинально меняться без особых трудозатрат со стороны программистов, сопровождающих данный программный продукт.

Заключение

Поскольку эксплуатация БД обычно длится в течение большого временного интервала, в период которого требуется обеспечить максимальную гибкость в обслуживании минимальной модификацией комплекса БД, разработка архитектуры интерфейсов занимает одну из ведущих ролей и требует применения заранее продуманных технологических приемов и методов. В частности, неизбежные модификации различных компонентов БД для исправления замеченных мелких ошибок и неточностей во время опытной эксплуатации программного продукта, проверки корректности данных, развития системы без заранее продуманной и подготовленной технологии осуществления модификаций будут либо невозможны, либо потребуют ресурсов, сопоставимых с ресурсами, затраченными на разработку всей системы. Успешная реализация и эксплуатация БД возможна только при комплексном, взвешенном подходе к решению основных подзадач. При этом с практической точки зрения эффективным подходом диктуется необходимость использовать современные формальные модели и технологический инструментарий.

Предлагаемый способ формального определения интерфейса приложения позволяет повысить производительность труда программистов, реализующих приложение. Такое формальное описание интерфейса является высокоуровневой непроцедурной документацией приложения. Использование формальных спецификаций создает хорошую основу для превращения приложения в легко модифицируемую надежную программную систему, разработанную на основе общепринятых стандартов.

БИБЛИОГРАФИЧЕСКИЙ СПИСОК

1. Extensible Markup Language (XML), W3C Proposed Recommendation 10-February-1998, REC-xml-19980210, T.Bray, et al, February 10, 1998.
2. XML Schema Part1: Structures, W3C Recommendation 02 May 2001. http://www.w3.org/ /TR/2001/REC-xmlschema-1-20010502/
3. Abrams M, Phanouriou C, Alan L.B., Stephen M.W., Shuster J.E. UIML: An Appliance-Independent XML User Interface Language // Computer Networks, Vol. 31, pp. 1695-1708.
4. XSL Transformations (XSLT), W3C Recommendation 16 November 1999. http://www.w3.org/TR/xslt.
5. *Артемов Д., Погульский Г., Альперович М.* Microsoft SQL Server 7.0 для профессионалов. Установка, управление, эксплуатация, оптимизация. М.: Издательский отдел «Русская редакция» ТОО “Channel Trading Ltd.”-1999.
6. *Касперски К.* Приемы защиты исходных текстов и двоичного кода // Открытые системы, № 7-8, 2001. С. 40-43.
7. *Аграновский А.В., Хади Р.А.* Защита Web-сценариев // Открытые системы, № 9, 2001. С. 55-57.