УДК 004.5

УНИВЕРСАЛЬНЫЕ ПРИНЦИПЫ ПРОЕКТИРОВАНИЯ ПОЛЬЗОВАТЕЛЬСКИХ ИНТЕРФЕЙСОВ

М. А. Артемов, А. А. Чиченин

Воронежский государственный университет

Поступила в редакцию 11.11.2011 г.

Аннотация. Рассмотрены ключевые принципы проектирования пользовательских интерфейсов и проблемы кроссплатформенности. Произведён обзор существующих решений для разработки кроссплатформенных приложений, описаны их сильные и слабые стороны. Изучено влияние аппаратных компонентов на пользовательский интерфейс. Определены требования к разработке современных сложных, многокомпонентных систем. Произведён обзор формата текстового описания универсальных пользовательских интерфейсов.
Ключевые слова: разработка кроссплатформенных приложений, универсальные пользовательские интерфейсы, среды разработки программного обеспечения, аппаратные компоненты пользовательского интерфейса.
Annotation. Reviewed keyuser interfaces design principles and problems of cross-platform development. Review of existing solutions for cross-platform application development, described pros and cons of these solutions. Investigated impact of hardware components on user interface. Defined requirements for modern complex and multicomponent systems development. Review of text format for defining universal user interfaces.
Keywords: Cross-platform application development, universal user interfaces, integrated development environments, user interfaces hardware components.

ВВЕДЕНИЕ

С развитием программирования и компьютерной техники вцелом, совершенствовались и графические интерфейсы. Увеличивались потребности в представлении новых видов информации и методов взаимодействия с ней. В операционных системах были реализованы встроенные, достаточно мощные графические подсистемы, со временем ставшие многоуровневыми, и большое число графических библиотек. Крайне важным моментом стало появление огромного числа носимых и портативных устройств, с всевозможными размерами и типами экранов и элементов управления.

Всё это привело к возникновению большого числа совершенно самостоятельных и несовместимых друг с другом платформ и технологий, с помощью которых сейчас создаются графические пользовательские интерфейсы.

В данный момент возникают задачи по созданию не только веб-сайтов компаний, но и связанных с ними по смыслу и набору информации клиентских приложений для различных операционных систем и мобильных платформ.

Современные требования к разработке комплексных систем. Как правило, при создании комплексной системы происходит разработка веб-сайта, в состав которого входит одно или несколько веб-приложений и клиентских приложений под несколько различных мобильных платформ. В некоторых случаях необходимо разработать ещё и прикладные клиентские приложения под различные операционные системы. Существующие решения не позволяют реализовывать такого рода системы используя единую систему проектирования и программирования.

Общие принципы. В процессе развития различных технологий для проектирования пользовательских интерфейсов большую популярность получила парадигма «МПК», или «Модель, Представление, Контроллер». В основном, причиной появления данной парадигмы послужили ограничения, накладываемые на веб-разработчиков тем фактом, что графический интерфейс находится в браузере пользователя, а модули, реализующие бизнес-логику — на удалённом сервере.

Большинство крупных компаний так или иначе использует данную парадигму в своих наиболее новых и перспективных разработках:

• Microsoft — в ASP.NET MVC [1], для создания веб-приложений. Данная технология должна заменить устаревающую систему шаблонизации в ASP.NET.

• Apple — в системе Cocoa, для создания как прикладных, так и мобильных приложений.

• Наиболее популярные и мощные системы для разработки веб приложений на языках Python и PHP (Django, Zend, CodeIgnitor) также используют данную парадигму как основу архитектуры.

В «МПК» различные компоненты системы строго отделены друг от друга. Инкапсуляция позволяет не только увеличить стабильность системы и увеличить её архитектурную целостность, но и явно отделить модули реализующие бизнес-логику и взаимодействие с внешними системами от модулей и компонетнов реализующих логику и отображение графического интерфейса.

Всё это, в конечном счёте, позволяет создавать группы приложений, совместно реализующих определённую крупную задачу и предоставляющую конечным пользователям наиболее подходящие для работы с определёнными её аспектами графические интерфейсы.

Кроме того, упомянутая ранее парадигма «МПК» может быть наложена на существующие методологии предпроектного анализа, проектирования и планирования. Что позволит стандартизировать и значительно ускорить процесс разработки сложных многокомпонентных систем.

Влияние аппаратных компонентов и элементов на графическую составляющую пользовательского интерфейса. В качестве базы, для проведения исследования были выбраны следующие конфигурации:

— Компьютеры, устанавливаемые на рабочих местах пользователей и использующих для взаимодействиямонитор, клавиатуру и манипулятор «мышь».

— Ноутбуки, в которых роль мыши выполняет «тачпад» (сенсорная панель) или аналогичное устройство-целеуказатель.

— Мобильные телефоны различных конфигураций (с аппаратной клавиатурой, только с «тачскрином» (графическая сенсорная панель), дополнительно требующие стилус (компьютерное перо).

— Дополнительно была рассмотрена возможность использования голосового управления устройствами и управления жестами при помощи веб-камеры.

Для проведения исследования было приглашено 10 человек, которым по-очереди предлагалось произвести один и тот же набор действий на различных устройствах. После чего им было предложено описать проблемы, возникшие у них в процессе. Кроме того за каждым тестирующим велось дополнительное наблюдение. Затем была произведена беседа в процессе которой совместно обсуждался опыт пользования различными устройствами, их плюсы и минусы.

В результате сравнения приложений, выполняющих одинаковые действия, но предназначенных для различных платформ было выявлено:

1. Основным фактором, влияющим на дизайн интерфейса приложения является диагональ экрана устройства и его разрешающая способность. Приложения, запускаемые на персональных компьютерах и ноутбуках с диагональю экрана большей 10 дюймов слабо отличаются на различных операционных системах. Тем не менее большинство приложений способно адаптироваться на широкий диапазон диагоналей от 10 до 32 дюймов.

2. Приложения одинакого хорошо совместимы как с манипуляторами «мышь» так и с «тачпадами». Хотя некоторые приложения (в особенности в операционной системе OSX) значительно лучше адаптированы под работу с последними, так как поддерживают многопальцевые жесты. Данную группу указателей иногда называют общим словом «целеуказатели».

3. Использование вместо целеуказателей «тачскрина» на мобильных платформах, и даже на экранах с большой диагональю и с поддержкой «тачскрина», тем не менее, достаточно затруднено. Приложения, предназначенные для работы с целеуказателем, в некоторых случаях оказывались не пригодными для использования, а в остальных — просто не удобными.

4. Клавиатура в большинстве современных приложений используется для ввода каких-либо данных, но не для навигации.

5. Для навигации удобнее всего пользоваться аппаратными кнопками, предусмотренными разработчиками устройства, так как они остаются на своих местах в независимости от используемого приложения. Даже при использовании устройства с другим расположением этих же кнопок пользователь адаптируется быстро.

При разработке универсального интерфейса необходимо решать проблемы:

1. Кроссплатформенности между программным обеспечением.

2. Совместимости между устройствами со схожей диагональю экрана, но различными аппаратными кнопками для навигации.

3. Минимизации дублирующегося кода, при разработки различных реализаций под различные размеры экрана.

4. Веб-приложения на мобильных платформах сильно ограничены как размером экрана, так и отсутствием доступа к большинству аппаратных элементов ввода-вывода устройства.

5. Веб-приложения слабо отличаются от прикладных, но как-правило работают с гораздо большей вертикальной областью видимости и для работы с ними пользователю приходится их прокручивать. Тем не менее, такой же подход уже используется и для мобильных приложений и не является проблемой.

6. Голосовое управление удобно для использования в автомобиле, как дополнение к основным функциям.

7. Управление жестами при помощи видеокамеры используется только в развлекательных приложениях.

Проблема кроссплатформенности решается при помощи конвертеров универсального формата описания интерфейсов в форматы, совместимые со всеми целевыми платформами.

Проблемы совместимости между устройствами и минимизации дублирующегося кода решаются в самом формате универсального описания интерфейсов, о котором более подробно рассказано далее.

Формат описания универсальных интерфейсов. Интерфейсы описываются набором объектов, хранящихся в формате JSON [2]. Объект состоит из наборов полей, определяющих его поведение и назначение. Базовыми сущностями являются: компоненты, пакеты компонентов, проекты. Для отдельного приложения создаётся проект, в состав которого включаются нужные компоненты и пакеты компонентов. Кроме того, в проекте можно использовать внешние пакеты компонентов. Это позволяет создать как базовые пакеты универсальных компонентов, которые можно использовать на различных платформах, так и пакеты, предназначенные для работы только на некоторых платформах.

Общая структура проекта представлена на рис. 1.

КЛАССЫ КОМПОНЕНТОВ

Объекты, из которых состоят интерфейсы делятся на два класса, в каждом из которых выделено по два подкласса:

1) универсальные компоненты:
— компоненты (базовые);
— сегрегаты;
2) специализированные компоненты:
— виджеты;
— виды.

Функции универсальных компонентов (кнопок, полей ввода и т.п.) определяются более сложными компонентами, в состав которых они

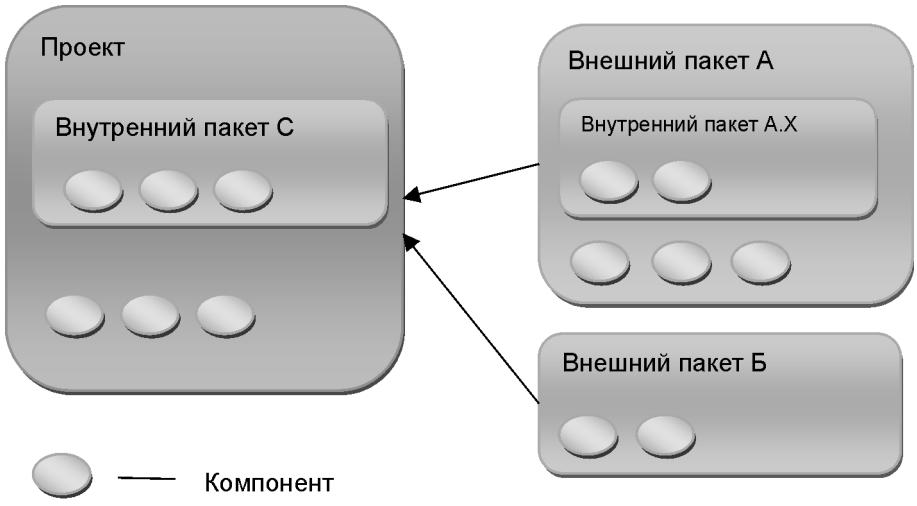

Рис. 1. Структура проекта

входят. Функции специализированных компонентов (например, диалогов выбора файла, шрифта или цвета), как правило, определяются в них самих и дополняются функциями, необходимыми в конкретном приложении.

Описанные классы компонентов можно изобразить на примере окна приложения, которое является видом (рис. 2).

Слияние различных компонентов для обобщения функциональности.

Одной из ключевых особенностей описываемой технологии является слияние. Слияние используется для описания новых компонентов интерфейса на основе уже существующих. При слиянии допускается не только усложнять существующие компоненты, но и упрощать их. Это позволяет очень гибко менять поведение компонента при необходимости. При слиянии допускается использование одного из следующих методов:

— Дополнение: в компонент добавляются поля целевого компонента, а если в текущем компоненте уже есть поля, которые необходимо добавить, то:

• если типы полей совпадают и значения являются объектами — происходит добавление полей целевого компонента в текущий компонент и его субкомпоненты;

• если типы полей совпадают и являются массивами, то массив текущего компонента дополняется элементами массива целевого компонента;

• в остальных случаях значение полей текущего компонента заменяется значениями полей целевого компонента;

— Перекрытие: добавление полей целевого компонента, если они отсутствуют в текущем компоненте;

— Вырезание: из текущего компонента поля удаляются по следующим правилам:

• если типы полей совпадают и значения полей являются объектами — происходит рекурсивное удаление полей текущего компонента, которые присутствуют в целевом компоненте;

• если типы полей совпадают и значения полей являются массивами — происходит удаление тех элементов массива текущего объекта, которые есть в массиве целевого компонента;

• в остальных случаях удаляются поля текущего компонента, присутствующие в целевом компоненте;

— Удаление: из текущего объекта удаляются поля текущего компонента, присутствующие в целевом компоненте;

Слияние производится последовательно, для всех правил хранящихся в поле «type». Если в объекте указано это поле — то слияние производится сразу после загрузки и определения пространства имён.

Сложное слияние может использоваться и в пакетах, включая пакеты проектов. При этом правила, указанные в поле «merge»пакета должны применяться ко всем компонентам, находящимся в этом пакете перед правилами для отдельных компонентов. Данная функциональность позволяет модифицировать определённое поле всех компонентов одного пакета и получить новый пакет, учитывающий, например, особенности специфической платформы.

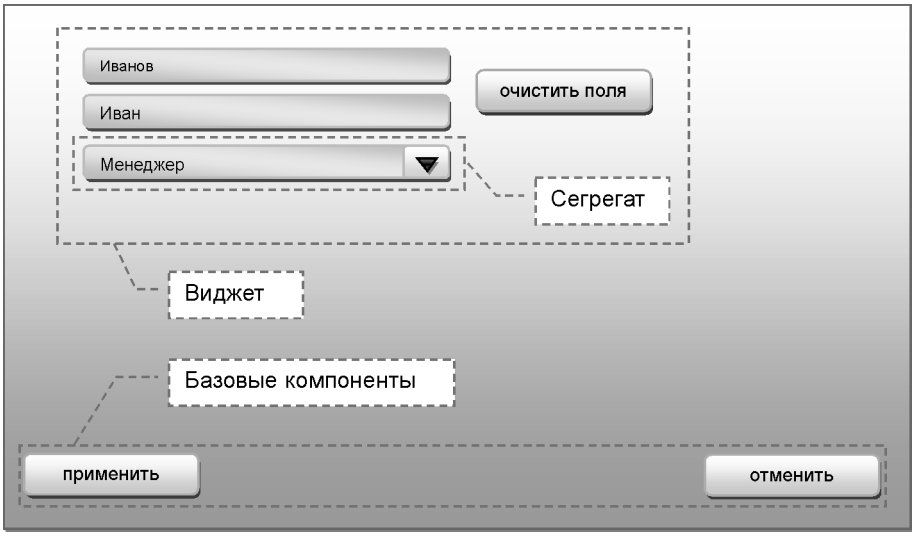

Рис. 2. Пример окна приложения (вид)

ЗАКЛЮЧЕНИЕ

На базе рассмотренного формата описания графических интерфейсов реализован набор базовых компонентов и библиотеки для работы с ними на языках Python [3] и Java. Создан прототип системы для хранения документации по описываемой теме.

Для среды разработки Eclipse [4] был создан прототип расширения [5], позволяющего работать с описанным форматом универсальных интерфейсов и управлять пакетами компонентов.

Артемов Михаил Анатольевич – заведующий кафедрой математического обеспечения и администрирования информационных систем Воронежского государственного университета, доктор физико-математических наук, профессор. E-mail: atremov_m_a@mail.ru

Чиченин Александр Александрович – аспирант кафедры программного обеспечения и администрирования информационных систем факультета прикладной математики информатики и механики Воронежского государственного университета. E-mail: achichenin@dataart.com

СПИСОК ЛИТЕРАТУРЫ

1. ASP.NET MVC Documentation / MicrosoftCorporation. — (http://www.asp.net/mvc).

2. 1&1 Internet, Inc. Introducing JSON / 1&1 Internet, Inc. — (http://json.org).

3. Python Software FoundationPython documentation / Python Software Foundation. — (http://docs.python.org).

4. The Eclipse Foundation Eclipse documentation / The Eclipse Foundation. — (http://www.eclipse.org/documentation/).

5. The Eclipse Foundation Platform Plugin Developer Guide / The Eclipse Foundation.— (http://help.eclipse.org/galileo/index.jsp).

Artemov Mikhail A. – Head of Department Software & Information System Administering, Voronezh State University, doctor of Physics-math. Sciences, Professor. E-mail: atremov_m_a@mail.ru

Chichenin Alexander A. – Post-Graduate Student, Department Software & Information System Administering, Voronezh State University. E-mail: achichenin@dataart.com