# Лекция № 2. Стандарты на интерфейсы. Средства проектирования пользовательских интерфейсов.

***Стандартизация пользовательского интерфейса.***

Ведущие специалисты в области человеко-машинных компьютерных систем уже в середине 70-х годов осознали необходимость формирования единых подходов к реализации пользовательского интерфейса.

В настоящее время солидные фирмы серьезно относятся к проблемам интерфейса. Существуют специальные организации, отвечающие за разработку стандартов.

Американский Национальный институт стандартов (ANSI) имеет по данному направлению специальную консультативную группу – Комитет по стандартам интерфейса «человек-компьютер» (The Human-Computer Interface Standard Committee).

Среди международных организаций можно назвать, например, Международный консультативный комитет по телеграфии и телефонии (International Telegraph and Telephone Consultation Committee), изучающий особенности интерактивных элементов интерфейса.

В 1987 г. (IBM) было положено начало создания единой среды разработки приложений - Systems Application Architecture (SAA). Данный проект предусматривал не только разработку единых принципов создания приложений, но и «материализацию» этих принципов на основе соответствующей технологической базы.

Проект SAA содержит 4 компонента:

- Соглашения по интерфейсу пользователя – CUA (Common User Access);
- Соглашения по программному интерфейсу – CPI (Common Programming Interface);
- Соглашения по разработке приложений – CA (Common Applications);
- Соглашения по коммуникациям – CCS (Common Communications Support).

В качестве технологической базы для реализации соглашений по пользовательскому интерфейсу было предложено конкретное инструментальное средство – Programming Toolkit для операционной системы OS/2.

Первоначально в разработке проекта SAA участвовали преимущественно фирмы IBM и Microsoft. Основные положения проекта воплотились корпорацией IBM применительно к OS/2, а фирмой Microsoft - в рамках семейства ОС Windows.

В марте 1997 г. фирма Microsoft выпустила пакетVisual Studio 97, в который вошли все созданные ею инструментальные средства разработки приложений, а также средства автоматизации сопровождения программных продуктов Visual SourceSafe.

***Стандарты.***

Хотя требования и спецификации, изложенные в CUA, пока юридически не стали международным стандартом, ориентация огромного числа производителей ПО на интерфейс MS Windows позволяет считать их стандартом де-факто.

Для Unix-систем аналогичный почти-стандарт представлен архитектурой Xwindow.

Поскольку разработка интерфейса является необходимой частью разработки любого программного обеспечения, основные требования к интерфейсам можно найти в стандартах, посвященных жизненному циклу программных средств. Примерами таких стандартов являются следующие:

MIL-STD-498 – стандарт разработки ПО Министерства обороны США;

ISO/IEC 12207:1995 («Процессы жизненного цикла программных средств»);

В России – комплект документов «Единая система программной документации» (появился в 1977 г. и постоянно корректируется).

***Руководящие принципы и инструкции.***

На основе существующих стандартов разрабатываются руководящие принципы и инструкции по разработке. *Принципы* содержат основополагающие требования. Инструкции относятся к элементам представления информации и взаимодействия и представляют собой правила и объяснения для создания элементов интерфейса и внешнего вида.

Руководящие принципы, отраженные в инструкциях, должны позволять пользователю применять к интерфейсу свое знание реального мира. Интерфейс должен иметь схожее поведение с объектами и метафорами реального мира.

***Нормативы.***

Нормативы затрагивают три области проектирования интерфейсов: физическую, синтаксическую и семантическую.

*Физическая* область относится к аппаратному обеспечению программного пользовательского интерфейса. Эти нормативы качаются расположения клавиш, их раскладки и проектирования, использования мыши, устройств рукописного ввода и т.п.

*Синтаксическая* область обобщает правила размещения информации на экране и последовательности действий пользователя (например, прямое манипулирование объектами).

*Семантическая* – раскрывает сущность элементов и действий, составляющих часть интерфейса (например, Exit – конец взаимодействия с диалоговым окном, выход из программы; Cancel – остановка любого незаконченного действия и возврат на шаг назад).

Инструкции и нормативы распределяются в порядке их важности по отношению к пользовательскому интерфейсу. Каждое руководство содержит разделы «когда используется» и «как используется». Проектировщики должны следовать всем инструкциям для обеспечения базисного уровня целостности интерфейса.

Решения о дополнительных составляющих нормативов должны приниматься на основе индивидуального или корпоративного стиля, распределения задач, ресурсов, бюджета.

Стандарты и руководящие принципы являются строительными блоками, на которые должны опираться все ваши усилия по разработке и проектированию. Однако строгое следование им не обязательно приведет к созданию удобного интерфейса. Разработка интерфейса – больше искусство, чем наука.

***Средства разработки ПИ.***

Технология построения ПИ и инструментальные средства, используемые для ее реализации, образуют единое целое. Очередной шаг в развитии любой изх этих составляющих дает толчок к дальнейшему развитию другой. Материальной же основой существования любого ПИ является перечень устройств ввода/вывода, доступных конечному пользователю.

Эволюция ПИ:

Перфокарты – нет интерфейса.

Символьно-цифровые устройства ввода/вывода – интерфейс на основе диалога (интерфейс не отделен от программы, «внутреннее управление интерфейсом»).

Языки СУБД обеспечили раздельное описание данных и средств работы с ними.

ООП – позволило явно отделить друг от друга компоненты приложения, реализующие функциональность и интерфейс.

Растровая графика – появление принципиально нового типа ПИ – GUI.

Средства визуальной разработки – позволяют создавать макет ПИ на основе технологии WYSIWYG.

Средства визуальной разработки были созданы практически для всех популярных языков программирования, а также для вновь появившихся. Все эти инструменты обладают двумя основными достоинствами: во-первых, существенно повышают производительность труда программиста, и, во-вторых, обеспечивают стандартизацию пользовательского интерфейса за счет использования однотипных базовых элементов. В результате, глядя на готовое приложение, практически невозможно определить, на каком языке и с помощью какого инструмента оно было создано.

Наиболее удачно реализованные инструменты визуального программирования позволяют не только формировать облик отдельных окон и диалоговых панелей, но и представлять в наглядной форме взаимосвязь между элементами пользовательского интерфейса.

Аналогичными возможностями обладают многие современные инструментальные средства на базе проблемно-ориентированных языков (редактор GUI в составе пакета MATLAB).

Как и до появления средств визуального программирования, особое место среди других проблемно-ориентированных систем разработки занимают СУБД. Применение в них технологии WYSIWYG позволило им практически сравняться по мощности и эффективности с универсальными инструментами разработки GUI-приложений. Кроме того, наличие в СУБД средств визуального представления инфологической модели данных позволяет создавать более корректную модель пользовательского интерфейса по сравнению с универсальными инструментами.

Интерфейс систем реального времени имеет целый ряд существенных особенностей. Для его построения используются, как правило, специализированные инструментальные средства. Они сформировались в результате слияния SCADA-систем (Supervisory Control and Data Acquisition system – система сбора данных и оперативного диспетчерского управления) и средств визуального программирования общего назначения на базе одного из универсальных языков. Такой симбиоз получил название HMI/SCADA-систем (или MMI/SCADA). В настоящее время такие инструментальные средства существуют практически для всех платформ, на базе которых разрабатываются системы реального времени.

Несмотря на огромные потенциальные возможности систем визуального программирования, они в большинстве своем обладают одним существенным недостатком – в них не предусмотрена поддержка проектирования, разработки и сопровождения создаваемых приложений как единого технологического процесса. Это обстоятельство зачастую негативно влияет как на уровень программного продукта в целом, так и на качество его пользовательского интерфейса. Осознание этого факта привело к тому, что разработчики инструментов стали дополнять их относительно самостоятельными компонентами, поддерживающими отдельные этапы жизненного цикла программных продуктов. В пакете Visual Studio фирмы Microsoft существует компонента для управления версиями программного продукта – SurfaceSafe.

Появились специализированные инструменты тестирования GUI-приложений. Наиболее мощным из них считается продукт Rational Performance Suite фирмы Rational Rose. Данное средство обеспечивает автоматическую генерацию тестов, имитирующих работу пользователя, а также регистрацию и анализ результатов тестирования приложения, прежде всего с точки зрения качества пользовательского интерфейса.

Тем не менее в инструментах визуального программирования поддержку получают в основном этапы жизненного цикла, относящиеся к разработке и реализации приложений, и в значительно меньшей степени – к этапам проектирования.

Указанного недостатка лишены так называемые CASE-системы (Computer Aided Software Engineering – компьютерное проектирование программного обеспечения). Обязательным атрибутом такой системы является автоматическая (или автоматизированная) генерация программного кода на основе спецификации.

Особенностью CASE-систем является поддержка практически всех этапов жизненного цикла создаваемого приложения:

- стратегическое планирование (описание целей, факторов, ресурсов; моделирование стратегии; формирование структуры плана и политики фирмы-разработчика);
- описание предметной области (описание объектов предметной области и отношений между ними; интеграция различных моделей предметной области);
- анализ возможностей реализации (анализ существующих проектов);
- определение требований (моделирование потоков данных; создание и анализ прототипов; контроль полноты и согласованности требований);
- системное проектирование (декомпозиция и сборка проекта, имитационное моделирование создаваемого приложения);
- программирование (генерация кода и анализ его метрических характеристик);
- тестирование (автоматическая генерация контрольных примеров, регистрация и анализ результатов тестирования);
- документирование (создание и сопровождение библиотеки спецификаций);
- сопровождение и управление проектом.

Т.о. применение CASE-систем способствует проектированию и реализации ПИ, обладающего требуемыми свойствами. Некоторые из таких систем имеют в своем составе компоненты, предназначенные специально для разработки ПИ создаваемого приложения. Таким продуктом является CASE/4/0 фирмы MicroTOOL GmbH. Он содержит «дизайнер диалогов», обеспечивающий создание и моделирование ПИ.

Вместе с тем, сами по себе CASE-системы достаточно сложны в освоении и использовании, поэтому эффективность их применения прямо пропорциональна сложности создаваемого продукта.

**Инструментальные средства создания ПИ**.

Рост количества и многообразия интерактивных приложений, а также расширение области их применения обусловили наличие двух тенденций:

Во-первых, все существующие инструменты создания приложений стали оцениваться помимо других критериев еще и с точки зрения их пригодности для создания ПИ определенного уровня.

Во-вторых, появились инструментальные средства, специально предназначенные для проектирования и реализации ПИ.

Инструментальные средства создания ПИ могут быть отнесены к одному из следующих ***классов***:

- системы управления ПИ (User Interface Management System - UIMS);
- инструментальные средства проектирования и разработки интерфейса (Interface Builder - IB);
- инструментальные средства разработки интерфейса (Tools&Toolkit – T&T);
- средства прототипирования интерфейса (Prototyping Tools - PT).

***Система управления пользовательским интерфейсом*** – это интегрированный набор средств, помогающих программисту в создании и управлении различными интерфейсами пользователя. Основной концепцией UIMS является идея разделения интерфейса и прикладной программы (ее функционально наполнения).

Как правило, UIMS состоит из двух частей: одна обеспечивает разработку интерфейса, а вторая – управление пользовательским интерфейсом в процессе его работы с приложением. Многие UIMS имеют собственный язык определения интерфейса для представления требуемого диалога и генератор, который автоматически создает необходимый код из исходного описания на этом языке. В идеале UIMS должна, с одной стороны, позволять создавать различные интерфейсы для работы с одним и тем же

приложением, а с другой – поддерживать один и тот же интерфейс для различных приложений.

К данному классу могут быть отнесены некоторые CASE-системы и некоторые системы типа HMI/SCADA.

Примеры UIMS-систем, доступных через Интернет (freeware):

| | | |
|---|---|---|
| Amulet | http://www.cs.cmu.edu/~amulet | (X, MS Windows) |
| SUIT | http://www.cs.virginia.edu/~suit | (платф.независ. – исп. д/обучения) |
| Thistle | http://www.ltg.ed.ac.uk/software/thistle/index.html | (Java) |
| WINTERP | http://www.cybertribe.com/mayer/wintwrp/ | (UNIX/X/Motif) |

***Инструментальные средства проектирования и разработки интерфейса***. Этот класс образуют средства, которые обеспечивают создание интерфейса определенного (стандартизированного) типа для различных приложений, функционирующих в соответствующей операционной среде. Примерами таких средств могут служить Visual C++ и Delphi для MS Windows, Tk/TCL для Xwindows, PhotonApplication Builder (Phab) для графической среды Photon microGUI операционной системы QNX.

Некоторые представители данного класса поддерживают только этап проектирования пользовательского интерфейса и ориентированы на совместное использование с одним из инструментов визуального программирования.

| | | |
|---|---|---|
| CanAda | http://wuarhive.wustl.edu/languages/ada/awtools/Canada/ | (MS Windows/Ada) |
| Forms | ftp://ftp.cs.ruu.nl/pub/SGI/FORMS/ | (SGI GL) |
| MotifGuide | http://www3.bc.sympatico.ca/GUIDE/ | (UNIX) |
| Visaj | http://www.ist.co.uk/visaj | (Java) |

***Инструментальные средства разработки интерфейса*** близки по своим характеристикам представителям предыдущего класса, но имеют либо более ограниченные функциональные возможности, либо представляют собой набор (библиотеку) элементов, на основе которых могут быть реализованы различные варианты GUI.

| | | |
|---|---|---|
| Action! | http://www.macromedia.com/Tools/Action/index.html | |
| Fresco | ftp://ftp.x.org/pub/R6untarred/xc/doc/hardcopy/Fresco | (C++/X/UNIX) |
| InterViews | ftp://interviews.stanford.edu | (C++/X/UNIX) |
| Qt | http://www.troll.no/qtinfo.html | (Windows, Linux, Unix и др.) |
| YACL | http://www.cs.sc.edu/~sridhar/yacl.html | (Windows, OS/2, X/Motif) |

***Средства прототипирования*** предназначены для построения макета (прототипа) ПИ и для сравнительной оценки альтернативных вариантов.

Для прототипирования могут применяться графические редакторы. Специалисты применяют средство InDesign, но отзывы о нем диаметрально противоположные.