Министерство образования Республики Беларусь
Учреждение образования
«Белорусский государственный университет
информатики и радиоэлектроники»

Кафедра интеллектуальных информационных технологий

В. В. Голенков, Н. А. Гулякина, Д. Г. Колб

ИНТЕЛЛЕКТУАЛЬНЫЙ ПОЛЬЗОВАТЕЛЬСКИЙ ИНТЕРФЕЙС

Рекомендовано УМО по образованию в области информатики и радиоэлектроники для специальности 1-40 03 01 «Искусственный интеллект» в качестве учебно-методического пособия

Минск БГУИР 2013

УДК 004.5:004.8
ББК 32.973.26-018.2+32.813
Г60

Р е ц е н з е н т ы:

кафедра информационно-вычислительных систем
учреждения образования «Военная академия Республики Беларусь»
(протокол №11 от 30.04.2012);

заведующий кафедрой информационных систем управления
факультета прикладной математики и информатики
Белорусского государственного университета,
доктор технических наук, профессор В. В. Краснопрошин

Голенков, В. В.

Г60 Интеллектуальный пользовательский интерфейс : учеб.-метод. пособие / В. В. Голенков, Н. А. Гулякина, Д. Г. Колб. – Минск : БГУИР, 2013. – 66 с. : ил.

ISBN 978-985-488-953-5.

Пособие содержит базовые теоретические сведения, необходимые для освоения курса «Интеллектуальный пользовательский интерфейс». Рассчитано на студентов, имеющих представление о теории операционных систем, теории объектно-ориентированного анализа и теории построения интеллектуальных систем.

Предназначено для студентов специальности «Искусственный интеллект» всех форм обучения.

УДК 004.5:004.8
ББК 32.973.26-018.2+32.813

ISBN 978-985-488-953-5

СОДЕРЖАНИЕ

ВВЕДЕНИЕ

Пособие содержит материалы лекций по курсу «Интеллектуальный пользовательский интерфейс» конспективного характера. Приведена информация о наиболее важных и теоретически наиболее научно проработанных проблемах и тенденциях развития традиционных и интеллектуальных пользовательских интерфейсов. Основное внимание уделяется аспектам логической и архитектурной организации пользовательского интерфейса. В пособии не затронуты вопросы, касающиеся дизайна пользовательского интерфейса, которые, без сомнения, являются важной составляющей любого прикладного программного обеспечения. Авторы рекомендуют темы, недостаточно усвоенные на основе приведенных в пособии материалов, закреплять, изучая первоисточники, приведенные в списке использованной литературы.

Авторы выражают благодарность студентам специальности «Искусственный интеллект» – Аврамовой Анастасии Игоревне и Грикян Ксении Валерьевне за оказанную помощь при подготовке пособия.

1. БАЗОВЫЕ ПОНЯТИЯ ТЕОРИИ ПОЛЬЗОВАТЕЛЬСКИХ ИНТЕРФЕЙСОВ

1.1. Взаимодействие «человек–компьютер»

Для чего необходимо изучение взаимодействия «человек–компьютер» (англ. human–computer interaction, HCI)? Изучение взаимодействия по линии «человек–компьютер» необходимо для совершенствования методов разработки, оценки и внедрения интерактивных компьютерных систем, предназначенных для использования человеком, а также в целях исследования различных аспектов этого использования.

В каких направлениях развивается HCI? HCI – полидисциплинарное научное направление, занимающееся изучением, планированием и разработкой взаимодействия между людьми (пользователями) и компьютерами. Зачастую HCI рассматривают как совокупность науки о компьютерах, бихевиоризма (раздел психологии, изучающий поведение), проектирования и других областей научного исследования.

Прикладной характер HCI выражается в том, что эта наука занимается:

– методологией и развитием проектирования интерфейсов (т.е., исходя из требований и класса пользователей, проектированием наилучшего интерфейса в заданных рамках, оптимизацией под требуемые свойства, такие, как обучаемость и эффективность использования);

– методами реализации интерфейсов (например, программными инструментариями, библиотеками и рациональными алгоритмами);

– методами для оценки и сравнения таких интерфейсов;

– разработкой новых интерфейсов и технологий взаимодействия;

– развитием описательных и прогнозируемых моделей, и теорией взаимодействия.

Долгосрочной задачей взаимодействия «человек–компьютер» является разработка системы, которая снизит барьер между человеческой когнитивной моделью того, чего он хочет достичь и тем, как компьютер понимает поставленные перед ним задачи.

Конечная цель изучения HCI – создание качественного человеко-компьютерного интерфейса – точки связи, мостика между человеком и компьютером.

HCI существует на стыке следующих научных направлений:

– компьютерная графика;

– инженерная психология;

– эргономика;

– теория организации;

– когнитивная наука;

– информатика.

1.2. Пользовательский интерфейс

Пользовательский интерфейс (ПИ) – это уровень системы «человек–компьютер», на котором происходит взаимодействие составляющих этой системы. ПИ включает в себя программное и аппаратное обеспечение (например, образы или объекты, отображаемые на экранах дисплеев), данные, полученные от пользователя посредством аппаратных устройств ввода (таких, как клавиатуры и мыши), и другие виды взаимодействий пользователя с крупными автоматизированными системами, такими, как воздушное судно и электростанция.

Пользовательский интерфейс включает в себя несколько аспектов:

– *область задач*: условия и цели, ориентированные на пользователя;

– *область машины*: среда, с которой взаимодействует компьютер, т.е. ноутбук студента в комнате в общежитии;

– *области интерфейса*: непересекающиеся области, касающиеся процессов взаимодействия человека и компьютера, но не относящиеся к сфере взаимодействия;

– *входящий поток*: поток информации, который начинается в области задач, когда пользователь имеет несколько задач, которые требуют использования компьютера;

– *выходной поток*: поток информации, который возникает в машине;

– *обратная связь*: узлы взаимодействия, проходящие через интерфейс (оцениваются, модерируются и подтверждаются, т.к. они проходят от человека через интерфейс к компьютеру и обратно).

Программный интерфейс пользователя ОС включает:

– средства отображения информации, отображаемую информацию и коды;

– командные режимы, язык «пользователь–интерфейс»;

– устройства и технологии ввода данных;

– диалоги, взаимодействие и транзакции между пользователем и компьютером, обратную связь с пользователем;

– поддержку принятия решений в конкретной предметной области;

– порядок использования программы и документацию на нее.

2. КЛАССИФИКАЦИЯ ПОЛЬЗОВАТЕЛЬСКИХ ИНТЕРФЕЙСОВ

2.1. Классификация пользовательских интерфейсов по формам взаимодействия и организации

По формам организации взаимодействия можно выделить следующие классы интерфейсов:

– *командный интерфейс*. Командный интерфейс называется так потому, что в этом виде интерфейса человек подает «команды» компьютеру, а компьютер их выполняет и выдает результат человеку;

– *WIMP-интерфейс* (Window – окно, Image – образ, Menu – меню, Pointer – указатель). Характерной особенностью этого вида интерфейса является то, что диалог с пользователем ведется не при помощи команд, а при помощи графических образов – меню, окон, других элементов;

– *SILK-интерфейс* (Speech – речь, Image – образ, Language – язык, Knowledge – знание). Этот вид интерфейса наиболее приближен к обычной, человеческой форме общения. В рамках этого интерфейса идет обычный «разговор» человека и компьютера.

По формам организации можно выделить следующие важные классы пользовательских интерфейсов:

– интерфейс командной строки;

– интерфейс меню;

– графический пользовательский интерфейс;

– объектно-ориентированный пользовательский интерфейс;

– тактильный интерфейс;

– жестовый интерфейс;

– материальный интерфейс пользователя;

– сенситивные интерфейсы;

– естественно-языковой пользовательский интерфейс;

– речевой пользовательский интерфейс;

– интеллектуальный пользовательский интерфейс.

2.2. Интерфейс командной строки и интерфейс меню

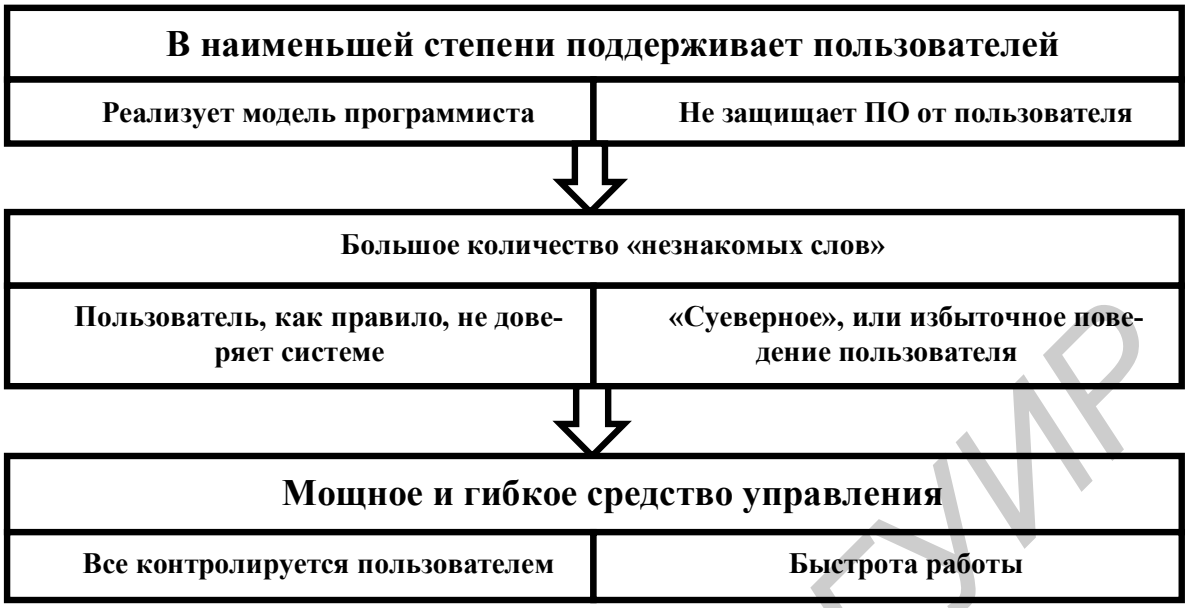

Рис. 2.1. Схема пользовательской модели интерфейса командной строки

Рис. 2.2. Схема семантики интерфейса командной строки

Достоинства:

– мощный и быстрый вид взаимодействия для опытных пользователей;

– гибкий интерфейс, простота комбинирования команд и параметров;

– взаимодействие, контролируемое самим пользователем;

– использование минимальной поверхности экрана;

– быстродействующий и эффективный интерфейс для знающих пользователей;

– возможность использования сокращенных названий команд;

– необходимость минимального объёма печати;

– возможность использования наряду с другими пользовательскими интерфейсами.

Недостатки:

– малое число (или полное отсутствие) подсказок и инструкций на экране;

– улучшения интерфейса не видны или не известны;

– как правило, требуется распечатанное руководство или электронная помощь;

– необходимо знание пользователями системы, программ и данных;

– как правило, отсутствует обратная связь и информация о состоянии выполнения задачи;

– необходимы навыки в наборе текста;

– необходимо запоминать команды и синтаксис;

– сложность в изучении;

– отсутствие смысла в названиях команд;

– сложность в понимании синтаксиса команд;

– необходимо точно следовать синтаксису команд;

– как правило, невозможно отладить команды и синтаксис.

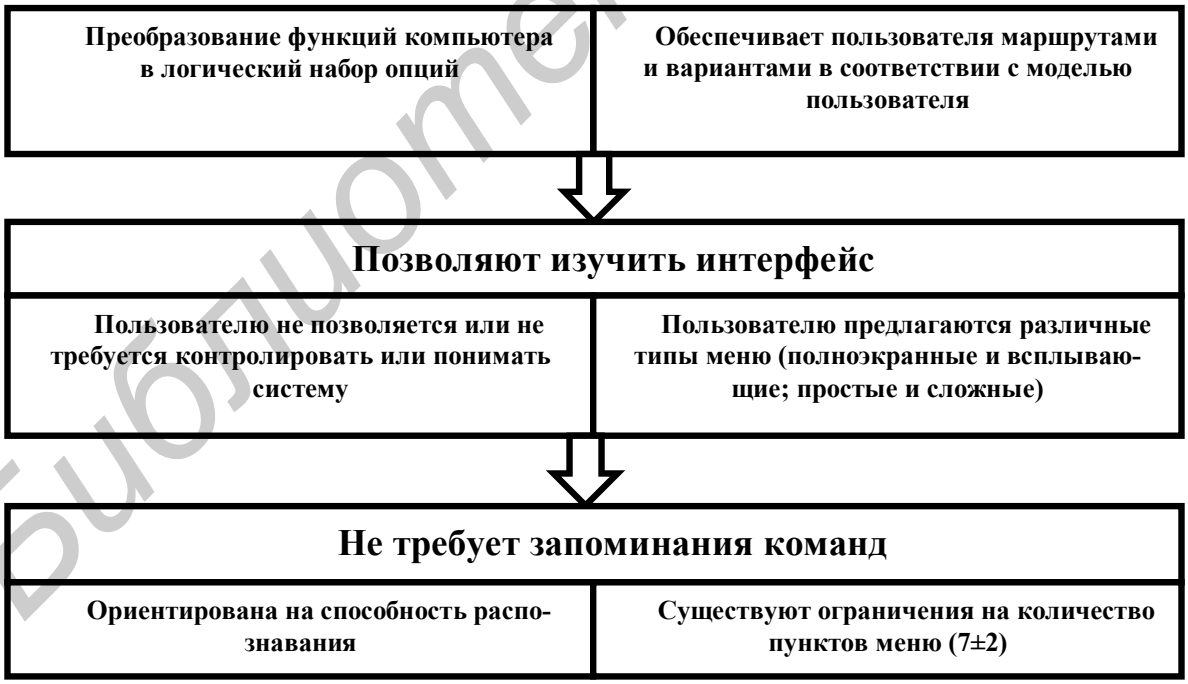

Рис. 2.3. Схема пользовательской модели интерфейса меню

Семантика интерфейса меню

Основная проблема: терминология, используемая в пунктах меню.

Пример: В одном меню встречаются: Exit, Quit, Escape, Close, Return, Back, Enter, Accept, Up, Down.

Некоторые команды по терминам очень похожи, но позволяют выполнять разные действия.

Достоинства:

– пользователям не надо помнить набор команды;

– отказ от использования клавиатуры уменьшает количество ошибок;

– навигация по иерархии проста для новых или случайных пользователей;

– сокращает время обучения пользователя;

– легко отслеживать ответы и корректировать ошибки;

– необходим минимальный набор печати;

– может быть использован совместно с другими типами интерфейсов;

– элементы меню и их расположения могут настраиваться пользователем;

– гибкая система выбора команд (мнемоники, мышь);

– поддерживает тип памяти, ориентированный на узнавание.

Недостатки:

– несоответствие или неэффективность для некоторых пользователей или задач;

– требуется техника навигации и выбора;

– не обязательно делает интерфейс удобным;

– занимает много места на экране;

– заставляет пользователя перемещаться через много уровней подменю;

– требует часто обновлять экран;

– пользователь должен понимать принцип группировки и иерархии меню;

– требует некоторых знаний о системе;

– пользователь может потеряться в иерархии меню;

– термины и наименования в меню непонятны пользователю;

– структура меню следует за синтаксисом «объект–действие»;

– синтаксис меню должен быть последовательным;

– использование режимов вынуждает пользователей следовать за системой.

2.3. Графический пользовательский интерфейс

Можно выделить следующие важные классы графических пользовательских интерфейсов:

– оконный интерфейс;

– WIMP (графический интерфейс);

– web-ориентированный интерфейс;

– индуктивный пользовательский интерфейс;

– масштабируемый интерфейс пользователя.

Оконный интерфейс – способ организации полноэкранного интерфейса программы, в котором каждая интегральная часть располагается в окне – собственном субэкранном пространстве, находящемся в произвольном месте «над» основным экраном. Может быть реализован как в текстовом виде (псевдографика), так и в графическом виде.

Web-ориентированный интерфейс – пользовательский интерфейс, который позволяет управлять различными программами через браузер. Web-интерфейсы удобны тем, что дают возможность вести совместную работу сотрудникам, не находящимся в одном офисе.

Индуктивный пользовательский интерфейс (ИПИ) – это новая модель пользовательского интерфейса, предлагающая способ сделать прикладные программы более простыми, разбивая функциональность на экраны или страницы, которые проще как описывать, так и понимать. Неофициальные тесты говорят о том, что в этой модели пользователи могут выполнять задачи так же быстро, как и в традиционных интерфейсах, и могут легче найти то, что им нужно. Цель ИПИ как нового способа проектирования программ – уменьшить объем размышлений пользователей, нужных для успешного перехода между частями продукта и использования его функций. Слово *inductive* («индуктивный») происходит от глагола *induce*, что означает – идти, следуя влиянию или убеждению. ИПИ является расширением обычного web-интерфейса.

Масштабируемый интерфейс пользователя – графический интерфейс пользователя, где рабочее пространство представляет собой большую или неограниченную плоскость, на которой расположены основные элементы, свойства и содержимое которых становится доступным по мере их «приближения» путём увеличения. Дальнейшее приближение содержимого

делает доступными более глубокие уровни. Таким образом, например, маленькая кнопка может иметь на себе, помимо основной надписи или значка, ещё и целую инструкцию по применению, которую может быть не видно при размере кнопки в 1 сантиметр, но которую можно с лёгкостью прочитать, если увеличить изображение кнопки. Примером такого менеджера является файловый менеджер среды Eagle Mode.

Характеристики проблемно-ориентированного пользовательского интерфейса:

– приложение состоит из иконки, первичного окна и вторичных окон;

– иконки представляют приложения или открытые окна;

– пользователи должны запустить приложение, прежде чем приступить к работе с объектами;

– содержание в основном показано с помощью текстовых полей со списками;

– обеспечивает пользователей функциями, необходимыми для выполнения задач;

– акцент делается на основную задачу по определению приложения;

– взаимосвязанные задачи поддерживаются другими приложениями;

– жесткая структура определяется функцией;

– пользователь не может прервать выполнение задачи;

– тренинг сфокусирован на приложении и его функциях;

– пользователи должны следовать структуре приложения;

– требуется много приложений – по одному на задачу.

Характеристики объектно-ориентированного пользовательского интерфейса:

– продукт состоит из набора взаимодействующих объектов или видов объектов;

– иконки представляют объекты, которыми можно манипулировать напрямую;

– пользователи открывают объекты в их представления на рабочем столе;

– папки и справочники являются визуальными контейнерами;

– обеспечивает пользователей запасами, необходимыми для выполнения задач;

– акцент делается на входные и выходные данные для объектов и задач;

– взаимосвязанные задачи поддерживаются использованием других объектов;

– гибкая структура определяется объектом;

– пользователь может прервать выполнение задачи;

– тренинг должен фокусироваться на общих концепциях, представлениях и функциях;

– пользователи могут выполнять задачи по-своему или совершенствовать процесс выполнения;

– мало объектов – больше повторного использования одних и тех же объектов во многих задачах.

Жестовый интерфейс – пользовательский интерфейс, позволяющий управлять программными и аппаратными средствами компьютера с помощью жестов. Разработкой занимаются Hitachi, Microsoft и др.

Материальный интерфейс пользователя – это разновидность интерфейса пользователя, в котором взаимодействие человека с электронными устройствами происходит при помощи материальных предметов и конструкций. Примером материального интерфейса можно назвать шаровой автоответчик Дюрелла Бишопа. Каждый шарик соответствует сообщению, оставленному на автоответчике. Перемещение шарика в специальную выемку воспроизводит связанное с ним сообщение или вызывает звонившего.

Тактильный пользовательский интерфейс – это разновидность интерфейса пользователя, в котором взаимодействие человека с электронными устройствами происходит при помощи прикосновения. Широко распространен в мобильных телефона, банкоматах.

Сенситивные интерфейсы – это разновидность интерфейса пользователя, в котором в ответ на взаимодействие человека с электронными устройствами происходит воздействие на органы чувств человека.

Пример: Телефон Samsung премиум-класса с сенсорным экраном Anycall Haptic (SCH-W420, SPH-W4200). Его отличает нестандартный пользовательский интерфейс, который использует зрение, слух и тактильные ощущения: нажав кнопку увеличения звука, вы услышите тикающий звук и почувствуете легкую вибрацию.

Естественно-языковой пользовательский интерфейс – это разновидность интерфейса пользователя, в котором взаимодействие человека с электронными

устройствами происходит посредством обмена сообщениями на естественном языке.

Речевой пользовательский интерфейс – это разновидность интерфейса пользователя, в котором взаимодействие человека с электронными устройствами происходит посредством обмена сообщениями с помощью голоса на каком-либо естественном языке.

Интеллектуальный пользовательский интерфейс (рис. 2.4) – по Д. А. Поспелову, это некоторая логическая машина, сложность которой гораздо выше обычной вычислительной машины и которая может выполнять следующие функции:

– общения;

– автоматического синтеза программы;

– обоснования;

– обучения.

Функция общения. Предполагается, что непрограммирующий пользователь будет общаться с ЭВМ на ограниченном естественном языке. Ограниченность языка состоит в том, что он используется для определенной цели – формулировки задач, которые должна решать ЭВМ. Его ограниченность проявляется не в объеме словаря, а скорее, в организации текстов, вводимых пользователем в ЭВМ. Важно, чтобы вводимый текст был понятен для ЭВМ.

Система общения, входящая в интеллектуальный интерфейс, – это не только система общения на основе текстовых сообщений, но и всевозможные системы ввода–вывода речевых сообщений, средства графического взаимодействия и средства типа курсора.

Функция автоматического синтеза программы. Сообщение пользователя должно преобразовываться в рабочую программу, которую ЭВМ может выполнить. Это заставляет иметь в составе интеллектуального интерфейса средства для реализации в ЭВМ процедур, которые обычно выполняет человек-программист. Для того чтобы это стало возможным, необходимо уметь перевести исходное сообщение пользователя на некоторый точный язык спецификаций, а затем породить из этой записи рабочую программу. Подобное преобразование требует специальных знаний, которые должны храниться в памяти ЭВМ.

Функция обоснования. У пользователя может возникнуть желание получить от ЭВМ обоснование решения. В функцию обоснования входит и функция объяснения, характерная для современных экспертных систем, и функция доверия, цель которой – повысить степень доверия пользователя к ЭВМ.

Функция обучения. Когда пользователь впервые подходит к ЭВМ, то он вправе ожидать, что сведения о работе с нею он сможет получить достаточно легко. Поэтому ЭВМ новых поколений снабжаются специальными средствами (тьюторами), с помощью которых пользователь постепенно постигает способы работы с ЭВМ и тонкости успешного общения с ней.

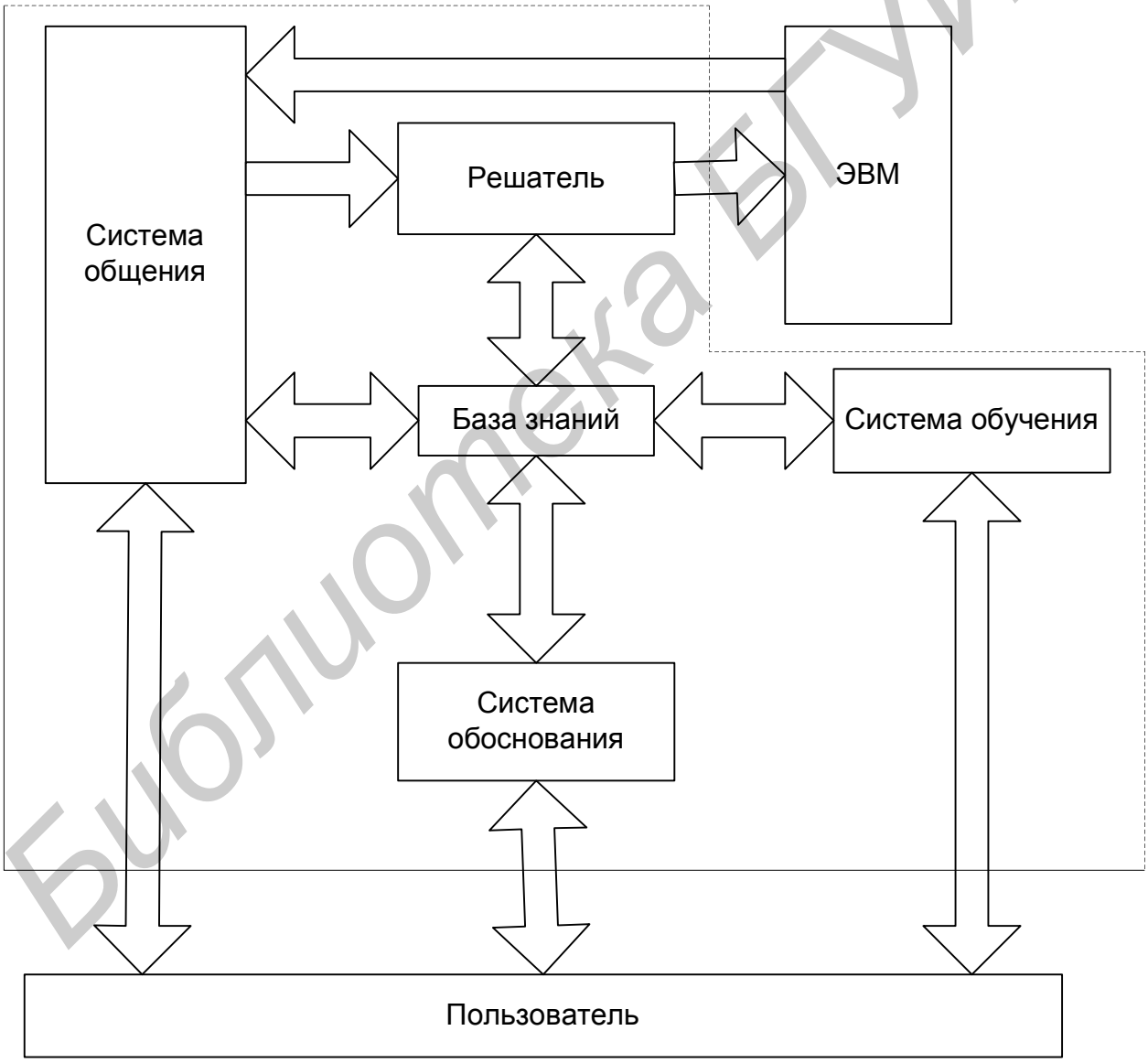

Рис. 2.4. Структура интеллектуального пользовательского интерфейса

3. КОГНЕТИКА И ЛОКУС ВНИМАНИЯ

3.1. Взаимодействие с пользовательским интерфейсом

Как мы взаимодействуем с пользовательским интерфейсом? Существует два способа взаимодействия:

Физическое взаимодействие. Руководства по разработке продуктов, взаимодействующих с нами физически, обычно содержат конкретную информацию, основанную на свойствах и возможностях человеческого скелета и органов чувств. Совокупность сведений в этой области составляет науку ***эргономику***.

Ментальное взаимодействие. «Мы должны овладеть эргономикой сознания, если мы хотим создавать интерфейсы, которые могли бы хорошо работать» (Дж. Раскин). Изучение прикладной сферы наших ментальных способностей называется когнитивным проектированием, или ***когнетикой***.

3.2. Эргономика

Эргономика (*согласно определению Международной Ассоциации Эргономики (IEA)*, *2010*) – это научная дисциплина, изучающая взаимодействие человека и других элементов системы, а также сфера деятельности по применению теории, принципов, данных и методов этой науки для обеспечения благополучия человека и оптимизации общей производительности системы.

Разделы эргономики:

Ми́кроэргономика (иногда её называют мини-эргономикой) – занимается исследованием и проектированием систем «человек–машина». В частности, проектирование интерфейсов программных продуктов находится в ведении микроэргономики.

Ми́ди-эргономика – занимается изучением и проектированием систем «человек–коллектив», «коллектив–организация», «коллектив–машина», «человек–сеть».

Ма́кроэргономика – исследует и проектирует более общие системы, такие, как «человек–общество», «организация–система организаций».

В 1986 г. проф. А. Е. Аствацатуровым был предложен термин «инженерная эргономика», разработаны методы и методологическая основа этого раздела эргономики.

Юзабилити (англ. Usability – удобопользование) – понятие в микроэргономике, означающее итоговый уровень удобности предмета для использования в заявленных целях. Международный стандарт ISO 9241-11 определяет *юзабилити* как «степень, с которой продукт может быть использован определёнными пользователями при определённом контексте использования для достижения определённых целей с должной эффективностью, продуктивностью и удовлетворённостью». При этом относительная важность всех трёх аспектов определяется этим самым контекстом.

Нормативные документы:

ISO 9241 – эргономические требования к офисной работе с визуальными терминалами (VDTs): Часть 11 – Руководство по юзабилити; Часть 110 – Принципы организации диалога.

ISO 9126 – качество программного продукта: Характеристики и подхарактеристики качества. Модель качества. Показатели качества в использовании.

CIF – формат описания юзабилити характеристик продукта и результатов юзабилити тестов.

Эргономические цели разработчика ПИ *(В. Андреев. "О чем надо помнить при разработке пользовательского интерфейса"(http://Usability.Ru)):*

Эффективность работы – означает обеспечение точности, функциональной полноты и завершенности при выполнении производственных заданий на рабочем месте пользователя.

Производительность работы – отражает объём затраченных ресурсов при выполнении задачи, как вычислительных, так и психофизиологических.

Удовлетворенность пользователя от работы – тесно связана с комфортностью его взаимодействия с приложением, и способствует сохранению профессиональных кадров на предприятии заказчика за счет привлекательности работы на данном рабочем месте.

3.3. Когнетика

Когнитивная психология (когнетика) – раздел психологии, изучающий когнитивные, то есть познавательные процессы человеческого сознания. Исследования в этой области обычно связаны с вопросами памяти, внимания, чувств, представления информации, логического мышления, воображения, способности к принятию решений.

Когнитивное бессознательное и когнитивное сознательное. Бессознательными называются те ментальные процессы, которые вы не осознаете в тот момент, когда они происходят. Для человеческого мозга обычным является процесс перехода от бессознательного к сознательному и наоборот.

Таблица 3.1

Свойства когнитивного сознательного и когнитивного бессознательного

Свойство	**Сознательное**	**Бессознательное**
Инициируется	Чем-то новым	Повторением
	Нестандартной ситуацией	Ожидаемыми событиями
	Опасностью	Безопасностью
Используется	В новых обстоятельствах	В привычных ситуациях
Решает задачи	Принятие решений	Работа с неветвящимися задачами
Принимает	Логические утверждения	Логические или противоречивые утверждения
Функционирует	Последовательно	Одновременно
Управляется	Волей	Привычными действиями
Производительность	Небольшая	Огромная
Период функционирования	Десятки секунд	Десятилетия (всю жизнь)

3.4. Внимание

Фокус внимания – волевое усилие, во время которого человек концентрируется на чем-то или ком-то сравнительно долгий момент времени. Человек может управлять фокусом внимания.

Локус внимания – волевое усилие, во время которого человек концентрируется на чем-то или ком-то сравнительно недолгий момент времени. Локус внимания в отличие от фокуса может часто меняться. Локусом внимания человек не может управлять.

По статистике психологов, *зрительные образы* обычно затухают через 200 мс, эта величина может варьироваться в пределах от 90 до 1000 мс; *слуховые образы* затухают в среднем через 1500 мс, эта величина может варьироваться в диапазоне от 900 до 3500 мс.

Восприятия не всегда откладываются в памяти. Большинство восприятий утрачиваются после того, как затухают. С точки зрения разработки интерфейсов из быстрого затухания сенсорных восприятий следует, что человек, прочитавший или услышавший пять секунд назад некоторое сообщение, необязательно сможет вспомнить его содержание. То есть сообщение должно оставаться на экране до тех пор, пока не перестанет быть актуальным (такой подход можно назвать наилучшим), или же необходимо предоставить пользователю возможность немедленно обработать эту информацию, прежде чем она исчезнет из его памяти. Когда некоторая информация становится локусом внимания, она перемещается в краткосрочную память. Там она будет храниться в течение 10 с.

Рис. 3.1. Обработка и запоминание информации человеком

Хранилище – сохраняет информацию (аудио, визуальную, тактильную), которая может быть достаточно объёмной и обладать высоким уровнем (7±2 предмета).

Краткосрочная память как область отвечает за процесс мышления и называется так же рабочей памятью. Имеет наименьшую пропускную способность.

Долгосрочная память – хранилище информации с неограниченной ёмкостью и продолжительностью хранения.

Формирование привычек. *Привычка* – действие, по мере повторения или с практикой для выполнения которого вы не задумываетесь.

Обратите внимание

1. Любая привычка означает отказ от внимания к деталям!

2. Привычки могут быть как полезными, так и вредными!

3. В случае идеального человекоориентированного интерфейса доля участия самого интерфейса в работе пользователя должна сводиться к формированию полезных привычек.

Одновременное выполнение задач. *Автоматичная задача* – любая задача, которую вы научились выполнять без участия сознания. (Та задача, которая не является автоматичной, естественно, находится непосредственно в локусе внимания.)

Интерференция – феномен психологии, когда при выполнении одновременно двух задач, ни одна из которых не является автоматичной, эффективность выполнения каждой из них снижается в результате конкуренции за область внимания.

Следовательно, когда вы повторяете какую-то последовательность действий, единственный способ предотвратить формирование привычки – это удерживать то, что вы делаете, в локусе внимания. Это очень сложно. Как обычно говорят, наше внимание «гуляет».

Пример удаления файла:

1. Задают вопрос: «Вы уверены?» После чего вам требуется ввести либо «Да», либо «Нет» в качестве ответа.

2. Так как ошибки случаются редко, вы обычно будете отвечать «Да» на любую команду, которая требует подтверждения.

3. В результате вы, не останавливаясь и не проверяя собственное намерение, вводите «Да».

Любой запрос о подтверждении, требующий установленного ответа, вскоре становится бесполезным. Разработчики, которые используют такого рода подтверждения, и администраторы, которые думают, что запросы о подтверждении обеспечивают безопасность, на самом деле не учитывают силу свойства формирования привычек, присущего когнитивному бессознательному.

Не существует идеального способа подтверждения операции. Даже если пользователь будет вводить обоснование удаления, – такой метод особенно

подходит для ситуаций, связанных с соблюдением правомочности действий, – это, в конце концов, приведет к тому, что он, то есть пользователь, станет каждый раз выбирать один и тот же стандартный ответ. Если основание для выполнения того или иного необратимого действия было с самого начала неверным, никакое предупреждение или запрос о подтверждении этого действия не поможет пользователю избежать ошибки.

Сингулярность (единственность) локуса внимания – способность человека «отстраиваться» от того, что отвлекает, необязательно проявляющаяся в виде реакций типа «все или ничего», как это было в предыдущих примерах. Реакция может быть пропорциональной уровню поглощенности или интенсивности помехи. По мере роста напряжения «люди все больше и больше концентрируются только на нескольких характеристиках окружающей обстановки, обращая все меньше и меньше внимания на другие». Если во время использования какого-либо интерфейса компьютер начинает работать неожиданным образом, то по мере того как ваше волнение, вызванное возникшей проблемой, возрастает, вы с меньшей вероятностью сможете замечать подсказки, сообщения и другие средства помощи пользователю.

Пример: Если вы напряженно работаете и слушаете музыку, то порой вы не замечаете не только, какая композиция исполняется, но и то, что звучит музыка. ***Вывод:*** Если вы сконцентрированы на чем-либо, то при возникновении критической ситуации вы можете музыку не услышать.

Эксплуатация единого локуса внимания. Если нам известно, где в данный момент внимание пользователя зафиксировано, мы можем производить изменения во всех остальных частях системы, зная, что они его не отвлекут. Это правило всегда используют фокусники. Человеку необходимо около 10 с для того, чтобы переключиться с одного контекста на другой или мысленно подготовиться к предстоящей задаче.

Возобновление прерванной работы. В нынешних системах, основанных на метафоре «рабочий стол», вы всегда должны самостоятельно осуществлять переход к необходимой задаче. Для интерфейса, который всегда возвращает вас туда, где вы остановились в последний раз, это был бы худший из возможных случаев, потому что если вы захотели бы вернуться к предыдущей задаче, вам вообще не пришлось бы совершать никаких действий.

Аналогичным образом, когда вы возвращаетесь на какой-либо сайт или web-страницу, лучше всего было бы возвратить вас на то место, где вы были последний раз, а не на начальную страницу; тем более если сайт хорошо разработан, то переход на начальную страницу осуществляется одним щелчком мыши. По этим же самым причинам очевидно, что когда вы открываете документ в каком-либо приложении, например в текстовом процессоре, вы должны вернуться к тому месту, где вы с ним работали в тот момент, когда закрыли или сохранили его в последний раз.

4. РЕЖИМЫ В ПОЛЬЗОВАТЕЛЬСКОМ ИНТЕРФЕЙСЕ. КВАНТИФИКАЦИЯ И УНИФИКАЦИЯ

4.1. Режимы

Режимы (modes) являются важным источником ошибок, путаницы, ненужных ограничений и сложности в интерфейсе. Тем не менее практика создания систем без режимов почему-то не находит широкого применения в разработке интерфейсов. Иногда так заманчиво добавить режим для того, чтобы получить такой же набор управления, какой уже есть, но с другими функциональными возможностями.

Жест (gesture) – это последовательность действий человека, которая выполняется автоматически (после старта). ***Пример жеста***: опытный пользователь набирает простое слово. Неопытный пользователь – буквы.

Формирование модуля (chunking) – объединение последовательности действий в жест, связанный с определенным психологическим процессом. То есть соединение отдельных элементов когнитивного процесса в единый ментальный модуль, что позволяет воспринимать множество элементов как целое.

Режимы проявляются в том, как реагирует интерфейс на тот или иной жест. Состояние интерфейса, при котором интерпретация данного конкретного жеста остается неизменной, называется *режимом*. Если жест получает другую интерпретацию, это значит, что интерфейс находится в другом режиме. Режимы создают проблемы по той причине, что привычные действия приводят к неожиданным результатам.

Из практических соображений необходимо использовать переключатели (radiobutton) вместо выключателей (checkbox).

Пример:

Переключатель: пользователь видит все варианты режима – мала вероятность ошибки.

Выключатель***:*** пользователь видит лишь один вариант режима, причем неизвестно какой – велика вероятность ошибки. За исключением случаев, когда текущее состояние находится в локусе внимания, – а обычно это не так, – выключатели могут приводить к ошибкам!

Ошибки, связанные с режимами:

– результаты недостаточной обратной связи, осуществляемой индикатором состояния системы;

– индикатор состояния системы не находится в локусе внимания пользователя.

Три метода предотвращения модальных (т. е. связанных с режимами) ошибок:

– не использовать режимы;

– обеспечить четкое различие между режимами;

– не использовать одинаковые команды в разных режимах, чтобы команда, примененная не в том режиме, не могла привести к неприятностям.

Следствие: только первый метод позволяет полностью избежать модальных ошибок; второй метод не всегда работает; третий метод не сокращает количество ошибок, но позволяет уменьшить их негативные последствия.

Крайний случай: можно привлекать внимание пользователя к индикатору текущего режима. Однако локус внимания пользователя будет переключаться с задачи на текущее состояние системы.

Модальные ошибки – это ошибки, связанные с тем, что пользователь неверно классифицирует или анализирует ситуацию. Термины «неверно классифицировать» и «анализировать» в данном случае указывают на активное, сознательное участие со стороны пользователя. Эти термины могу применяться до тех пор, пока он еще не в полной мере знаком с той или иной командой, но не тогда, когда применение этой команды стало для него автоматическим.

Определение модального интерфейса: интерфейс «человек–машина» является модальным по отношению к данному жесту, если, во-первых, текущее состояние интерфейса не находится в локусе внимания пользователя и, во-вторых, если в ответ на некоторый жест интерфейс может выполнить одно из нескольких возможных действий в зависимости от текущего состояния системы.

Истинно немодальный интерфейс – интерфейс, немодальный для любого жеста. Введем понятие степени модальности интерфейса Q.

Пусть Ni – некоторый немодальный жест, тогда $p(Ni)$ – вероятность применения данного немодального жеста, которая вычисляется для данного пользователя или в среднем для группы пользователей.

Значение Q берем из диапазона [0,1].

Если $Q = 0$, то интерфейс полностью модальный.

Если $Q = 1$, то интерфейс полностью немодальный.

Q будем рассчитывать по следующей формуле:

$$Q = \sum_i p(Ni).$$

Пример: *Рассмотрим интерфейс текстового редактора и жест нажатия клавиши <Backspace>. По второй части определения жест – модальный, так как то, какой именно символ удаляется, зависит от того, какой символ был введен последним, т.е. содержание характеризует состояние системы. Согласно первой части определения, данный жест – не модальный, так как мы осознаем, что удаляемый объект находится в локусе нашего внимания.*

Вывод: *жест – не модальный по отношению к интерфейсу текстового редактора.*

Диапазон (range) жеста g – набор состояний, в которых жест *g* имеет конкретную интерпретацию.

Пример:

Жест g вызывает действие a в режиме A.

Жест g вызывает действие b в режиме B.

Включен режим B. Необходимо выполнить действие a.

Необходимо: перейти в режим A, затем использовать жест g для выполнения действия a.

Следствие: разделение интерфейса на ограниченные области является неизбежным следствием наличия режимов. Группировка команд по разным диапазонам позволяет понять и научиться использовать сложные интерфейсы. Тем не менее существует возможность организовывать интерфейсы, которые могли бы создавать меньше ограничений, чем режимы. *Полностью человекоориентированный интерфейс должен состоять только из одного диапазона.*

Макрос – набор сохраняемых команд, который может быть выполнен одним жестом. Макросы используются, когда интерфейс управляется внешней программой. Макрос не может установить переключатель в какое-либо из его допустимых состояний, если сам макрос сначала не задаст системе вопрос о ее текущем состоянии.

Одно из возможных решений – обеспечить установку переключателя, имеющего несколько положений, в некое начальное состояние сразу после каждого случая его использования. Целесообразно предусматривать не более пяти состояний переключателя. В результате этого подсчет числа переключений

позволит всегда определить текущее состояние переключателя. Другим решением может быть применение набора переключателей.

Режимы могут лишать пользователя возможности взаимодействия с системой. Некоторые разработчики считают, что работа в жестких рамках установленной последовательности действий оказывается полезной, так как позволяет системе самой «направлять» действия пользователя. При некоторых обстоятельствах важно, чтобы пользователь принял то или иное конкретное решение к какому-то моменту времени или перед выполнением следующего шага из последовательности. Если же пользователь не имеет возможности принять свое решение, то и необходимости диалога с ним нет.

Возможные решения: запросы и подсказки должны даваться не модально, а так, чтобы пользователь мог в максимальной степени сохранять контроль над системой.

Пользовательские настройки – это такие изменения в программном обеспечении, которые не отражаются в документации. При использовании пользовательских настроек невозможно протестировать качество интерфейса или написать документацию для системы, так как её конфигурация разработчикам не известна заранее.

Основные минусы пользовательских настроек:

Потеря времени. Время, которое тратится на изучение и выполнение пользовательских настроек, большей частью является потерянным с точки зрения текущей задачи.

Психологически положительное восприятие пользовательских настроек. Наблюдения, сделанные во многих экспериментах, показывают, что интерфейс, который оптимизирует продуктивность, не обязательно является интерфейсом, который оптимизирует субъективные оценки.

Вывод относительно режимов: если вы разрабатываете модальный интерфейс, учитывайте, что пользователи будут всегда совершать модальные ошибки за исключением тех случаев, когда значение состояния, контролируемого данным режимом, находится в локусе внимания пользователя (и он может его видеть) либо в его кратковременной памяти. Задача разработчиков – показать, что данный режим используется правильно или что преимущества данного режима перевешивают его неизбежные недостатки.

Квазирежим (пружинный режим, режим с запертой пружиной) – включение и физическое удерживание того или иного элемента управления во время выполнения другого действия. Квазирежимы являются весьма эффективными с точки зрения устранения режимов. Для сохранения эффективности число квазирежимов, скорее всего, должно быть от 4 до 7. Существует *два типа сигналов*, которые вводятся в компьютер: те, которые служат *для создания содержания*, и те, которые используются *для управления системой*. С точки зрения практического подхода квазирежимы должны использоваться для управленческих функций, тогда как для создания содержания должны применяться операции без задействования квазирежимов. Именно этот подход можно считать наиболее верным, поскольку управлять системой, удерживая кнопку для включения квазирежима, труднее. Однако важнее, чтобы пользователь мог больше времени уделить созданию содержания и работе с ним, а не применению различных команд для управления системой.

Модели последовательности действий:

Существительное–глагол: сначала происходит выделение существительного (объекта), а затем выбирается глагол (операция).

Глагол–существительное: Сначала выбирается глагол (операция), а потом происходит выделение существительного (объекта).

Преимущества модели существительное-глагол с точки зрения локуса внимания пользователя:

– уменьшение количества ошибок;

– скорость;

– простота и обратимость.

Уменьшение количества ошибок. Последовательность глагол–существительное устанавливает режим. При использовании модели существительное–глагол команды выполняются сразу, пока еще находятся в локусе внимания пользователя.

Скорость. Пользователю не требуется переключать свое внимание с содержания (которое и вызвало необходимость выполнения операции) к самой команде и затем опять к содержанию, чтобы можно было выделить необходимый участок текста. В модели существительное–глагол вы сначала выделяете текст, находящийся в локусе вашего внимания, и затем переключаете внимание

на команду. Таким образом, число переключений локуса внимания уменьшается на единицу.

Простота и обратимость. При использовании модели глагол–существительное должна быть предусмотрена возможность отмены или отката команды. В модели существительное–глагол, для того чтобы выбрать другой текст, не требуется нажимать кнопку отката или применять какой-либо другой способ отмены действия.

Ряд исследований показывает, что в целом подход «существительное–глагол» является более предпочтительным. Применение методов «глагол–существительное» должно ограничиваться только выбором из палитр, если они предназначены для непосредственного использования.

Видимый элемент интерфейса. Элемент интерфейса можно считать видимым, если он в данный момент доступен для органов восприятия человека, либо был настолько недавно воспринят, что еще не успел выйти из кратковременной памяти пользователя. Для нормальной работы интерфейса «должны быть видимы только необходимые вещи – те, что идентифицируют части работающих систем, и те, что отображают способ, которым пользователь может взаимодействовать с устройством. Видимость отображает связь между предпринимаемыми действиями и их реальной отдачей».

Признаки наличия невидимых элементов:

– если интерфейс вынуждает вас запоминать существование того или иного его элемента, это означает, что этот элемент является невидимым;

– если вы вынуждены углубляться в недра интерфейса, чтобы случайно или в результате собственной настойчивости натолкнуться на последовательность действий, которая активизирует какой-то его элемент, то этот элемент является невидимым;

– если вам приходится обращаться за помощью к справочной системе для того, чтобы узнать, как выполнить то или иное действие, это означает, что методы выполнения этого действия невидимы.

Компьютерные игры – это недокументированные интерфейсы. Способы управления или пути достижения желаемых результатов в компьютерных играх невидимы. Стоит добавить к этим играм документацию, и они станут неинтересными. Однако большинство людей не хотят играть в игры, когда им требу-

ется выполнить свою работу, поэтому перед разработчиком интерфейсов стоит задача сделать каждый элемент своего продукта видимым.

Причины разнообразия методов решения задачи в одном интерфейсе:

– одни пользователи могут предпочесть один метод, а другие пользователи – другой;

– один метод (например, выделение, вырезание и вставка) может быть полезным для работы с разнесенными частями документа, а другой метод (выделение и перетаскивание) может быть эффективным, только когда на дисплее видны и исходная и конечная позиции одновременно;

– обратная совместимость – каждый из методов порожден традицией, и поэтому разработчики считают разумным использовать как можно больше навыков из уже существующих у пользователей;

– «нерешительность разработчиков» – проблема наличия каких-то двух и более вариантов, ни один из которых не имеет преимуществ по сравнению с другими, предусмотрительно «решалась» с помощью внедрения сразу всех этих вариантов.

Монотонный интерфейс – интерфейс, имеющий только один способ выполнения той или иной задачи. Чем более монотонным является интерфейс с точки зрения пространства данной задачи, тем легче для пользователя сформировать автоматичность ее выполнения, которая, в конце концов, создается тем, что снимается необходимость выбирать способ достижения результата.

Преимущества монотонных интерфейсов:

– интерфейс, который не имеет режимов и является – насколько это возможно – монотонным, был бы чрезвычайно удобным в использовании при условии, что все другие характеристики имеют по крайней мере нормальное качество, принятое для современных интерфейсов. Пользователь не тратит время на выбор метода решения задачи;

– использование продукта, интерфейс которого основан на немодальности и монотонности, могло бы быстро вызывать привыкание (близкое к зависимости) у пользователей, приводя к тому, что все они могут приобрести преданность этому продукту и предпочитать его всем другим.

Дихотомия «новичок–эксперт». Современные графические пользовательские интерфейсы, имеющие рабочий стол, являются сочетанием по крайней мере двух разных интерфейсов: с одной стороны, сравнительно видимой и бы-

строизучаемой системы на основе меню, а с другой – неполного собрания трудноизучаемых и труднозапоминаемых комбинаций клавиш. Сочетание двух ошибок не может дать правильного ответа.

Почему необходимо исключать дихотомию «новичок–эксперт»? Фаза изучения работы элемента интерфейса требует сознательного внимания. Поэтому простота и ясность функции, а также ее видимость имеют особую важность. Фаза профессионального владения интерфейсом главным образом характеризуется бессознательным использованием его возможностей. Такое использование подкрепляется следующими качествами интерфейса: соответствие задаче, немодальность и монотонность. Таким образом, хорошо разработанный, человекоориентированный интерфейс совсем не требует разделения на подсистемы для новичков и экспертов**.**

4.2. Квантификация

Количественный анализ интерфейса. Количественные методы часто могут свести спорные вопросы к простым вычислениям. Они помогают понять важнейшие аспекты взаимодействия человека с машиной. Существует множество методов количественного анализа элементов интерфейса. Рассмотрим модель GOMS, критерий эффективности Раскина, закон Хика и закон Фитса.

Модель GOMS – «правила для целей, объектов, методов и выделения» (the model of goals, objects, methods, and selection rules). Моделирование GOMS позволяет предсказать, сколько времени потребуется опытному пользователю на выполнение конкретной операции при использовании данной модели интерфейса.

Модель скорости печати GOMS – модель, основанная на оценке скорости печати. К формальному анализу, конечно, прибегают в случаях, когда необходимо выбрать один из двух вариантов разработки, когда даже небольшие различия в скорости могут давать большой экономический и психологический эффект. Иногда пользуются расширенными моделями GOMS, как, например, анализ с использованием метода критического пути GOMS (critical-path method GOMS, CPM-GOMS) или версия, называемая естественным языком GOMS (natural GOMS language, NGOMSL), в которой учитывается поведение неопытного пользователя.

Время, требующееся для выполнения какой-то задачи системой «пользователь–компьютер», – сумма всех временных интервалов, которые по-

требовались системе на выполнение последовательности элементарных жестов, составляющих данную задачу. Для большей части сравнительного анализа задач, включающих использование клавиатуры и графического устройства ввода, вместо проведения измерений для каждого отдельного пользователя можно применить набор *стандартных интервалов (табл. 4.1)*.

Таблица 4.1

Стандартные временные интервалы со средними значениями времени

Номенклатура	Пояснение
$K = 0{,}2$ с	*Нажатие клавиши. Время, необходимое для того, чтобы нажать клавишу*
$P = 1{,}1$ с	*Указание. Время, необходимое пользователю для того, чтобы указать на какую-то позицию на экране монитора*
$H = 0{,}4$ с	*Перемещение. Время, необходимое пользователю для того, чтобы переместить руку с клавиатуры на ГУВ или с ГУВ на клавиатуру*
$M = 1{,}35$ с	*Ментальная подготовка. Время, необходимое пользователю для того, чтобы умственно подготовиться к следующему шагу*
R	*Ответ. Время, в течение которого пользователь должен ожидать ответ компьютера*

Данные значения позволяют сделать правильную сравнительную оценку между двумя интерфейсами по уровню эффективности их использования, несмотря на то что абсолютные значения могут существенно отличаться от указанных.

Вычисления в модели GOMS начинаются с перечисления операций из списка жестов модели GOMS, которые составляют это действие. Затем идет определение точек, в которых пользователь остановится, чтобы выполнить бессознательную ментальную операцию, – интервалы ментальной подготовки, которые обозначаются символом *M*.

Основные правила, позволяющие определить, в какие моменты будут проходить ментальные операции:

– начальная расстановка операторов *M*;

– удаление ожидаемых операторов *M*;

– удаление операторов *M* внутри когнитивных единиц;

– удаление операторов *M* перед последовательными разделителями;

– удаление операторов *M*, которые являются прерывателями команд;

– удаление перекрывающих операторов *M*.

Начальная расстановка операторов М. Операторы *М* следует устанавливать перед всеми операторами *К* (нажатие клавиши), а также перед всеми операторами *Р* (указание с помощью ГУВ), предназначенными для выбора команд; перед операторами *Р*, предназначенными для указания на аргументы этих команд, ставить оператор *М* не следует.

Удаление ожидаемых операторов М. Если оператор, следующий за оператором *М*, является полностью ожидаемым с точки зрения оператора, предшествующего *М*, то этот оператор *М* может быть удален. Например, если вы перемещаете ГУВ с намерением нажать его кнопку по достижении цели движения, то в соответствии с этим правилом следует удалить оператор *М*, устанавливаемый по первому правилу. В этом случае последовательность *Р М К* превращается в *Р К*.

Удаление операторов М внутри когнитивных единиц. Если строка вида *М К М К М К*... принадлежит когнитивной единице, то следует удалить все операторы М, кроме первого. *Когнитивной единицей* является непрерывная последовательность вводимых символов, которые могут образовывать название команды или аргумент. ***Например***, *Y, перемещать, Елена Троянская или 4564,23* являются примерами когнитивных единиц.

Удаление операторов М перед последовательными разделителями. Если оператор *К* означает лишний разделитель, стоящий в конце когнитивной единицы (например, разделитель команды, следующий сразу за разделителем аргумента этой команды), то следует удалить оператор *М*, стоящий перед ним.

Удаление операторов М, которые являются прерывателями команд. Если оператор *К* является разделителем, стоящим после постоянной строки (например, название команды или любая последовательность символов, которая каждый раз вводится в неизменном виде), то следует удалить оператор *М*, стоящий перед ним. Если оператор *К* является разделителем для строки аргументов или любой другой изменяемой строки, то оператор *М* следует сохранить перед ним.

Удаление перекрывающих операторов М. Любую часть оператора *М*, которая перекрывает оператор *R*, означающий задержку, связанную с ожиданием ответа компьютера, учитывать не следует.

Зачем нужно измерять эффективность интерфейса? Модель GOMS позволяет определить время, необходимое пользователю на выполнение любой

четко сформулированной задачи, для которой данный интерфейс предусмотрен. Модели анализа не могут дать ответ на вопрос о том, насколько быстро должен работать интерфейс.

Определение информации (в техническом смысле) – квантификация некоторого объема данных, передаваемых с помощью средства коммуникации, как, например, при разговоре двух людей по телефону, или если человек подает некоторый сигнал машине, например с помощью нажатия кнопки ГУВ, когда курсор находится в определенной области экрана.

Информационно-теоретическая производительность определяется так же, как понятие производительности определяется в термодинамике – отношением мощности на выходе к мощности на входе процесса.

Информационная производительность интерфейса *E* определяется как отношение минимального количества информации, необходимого для выполнения задачи, к количеству информации, которое должен ввести пользователь. *E* изменяется в пределах от 0 до 1. Если никакой работы для выполнения задачи не требуется или работа просто не производится, то производительность составляет 1. Производительность *E* равняется 0 в случаях, когда пользователь должен ввести информацию, которая совершенно бесполезна.

Количество передаваемой информации (n – количество равновероятных вариантов)	$\log_2 n$	
Количество информации для каждого варианта	$\frac{1}{n}\log_2 n$	(4.1)
Если вероятности каждого варианта не равновероятны, то количество информации для одного варианта определяется (*i*-я альтернатива имеет вероятность *p(i)*)	$p(i)\log_2(\frac{1}{p(i)})$	(4.2)

Определение количества информации: количество информации является суммой (по всем вариантам) выражения (4.2), которое при равновероятных вариантах сводится к выражению (4.1). Следствие: информационное содержание интерфейса, в котором возможно сделать только нажатие единственной клавиши (а ненажатие клавиши не допускается), составляет 0 битов.

$$1 \times \log_2(1) = 0 .$$

4.3. Закон Фитса и закон Хика

Различные количественные законы, имеющие отношение к разработке интерфейсов, имеют хорошее когнитивное обоснование. Эти законы часто дают дополнительные данные, на основе которых можно принимать те или иные решения, связанные с разработкой интерфейсов.

Закон Фитса позволяет определить количественно тот факт, что *чем дальше находится объект от текущей позиции курсора или чем меньше размеры этого объекта, тем больше времени потребуется пользователю для перемещения к нему курсора.*

Рис. 4.1. Иллюстрация к закону Фитса для одномерного случая

Понятие дистанции: пусть кнопка является целью некоторого перемещения. Тогда под дистанцией будем понимать длину прямой линии, соединяющей начальную позицию курсора и ближайшую точку целевого объекта.

Закон Фитса для одномерного случая: ***Время (мс)***$= a + b\log_2(\frac{D}{S}+1).$

Пусть размер объекта вдоль линии перемещения курсора обозначается как *S*, а дистанция – от начальной позиции курсора до объекта *D* (см. рис. 4.1).

Константы *a* и *b* устанавливаются опытным путем по параметрам производительности человека. Для приближенных вычислений можно использовать следующие значения: $a = 50$, $b = 150$.

Пояснения к закону Фитса:

–- вычисляемое время отсчитывается от момента, когда курсор начинает движение по прямой линии, до момента, когда пользователь щелкает мышью по целевому объекту;

– логарифм по основанию 2 является мерой трудности задачи в количестве битов информации, которое требуется для описания (одномерного) пути перемещения курсора;

– для вычисления времени можно использовать любые единицы измерения дистанции, так как *D/S* является отношением двух дистанций и поэтому не зависит от единицы измерения;

– закон Фитса может применяться только к таким перемещениям, которые невелики относительно размеров человеческого тела и которые являются непрерывными (совершаемыми одним движением).

Закон Хика позволяет количественно определить наблюдение, заключающееся в том, что *чем больше количество вариантов заданного типа вы предоставляете, тем больше времени требуется на выбор*.

Формальная запись закона Хика

Если все варианты равновероятностные:

Время (мс) $= a + b\log_2(n+1).$

Если вероятность i-го варианта равна p(i):

Время (мс) $= a + b\sum_i p(i)\log_2(\frac{1}{p(i)}+1).$

Константы *a* и *b* в большой степени зависят от многих условий, включая то, как представлены возможные варианты, и то, насколько хорошо пользователь знаком с системой. Если варианты представлены непонятным образом, значения *a* и *b* возрастают. Наличие навыков и привычек в использовании системы снижает значение *b*. При отсутствии более точных данных для проведения быстрых и приблизительных вычислений можно воспользоваться теми же значениями *a* и *b*, которые использовали для закона Фитса.

Практическое следствие из закона Хика: при любых положительных и ненулевых значениях *a* и *b* предоставление пользователю сразу нескольких вариантов одновременно обычно является более эффективным, чем организация тех же вариантов в иерархические группы; выбор из одного меню, состоящего из восьми элементов, производится быстрее, чем из двух меню, состоящих из четырех элементов каждое.

4.4. Унификация

Зачем нужна унификация? Разные приложения имеют разные наборы команд, и пользователь обычно не может в целом использовать команды приложения А при работе с приложением В или наоборот. Если же сделать команды независимыми от приложений, то тем самым мы сможем устранить модальность, которая изначально им присуща.

Аспекты унификации (элементарные действия):

– пользователь не обращает внимание на физические действия, которые выполняет при работе на компьютере, ввиду их однообразности;

– при использовании любых приложений требуется вводить какой-то текст.

Некоторые выводы:

– для создания человекоориентированного интерфейса для компьютеров или компьютерных систем важным шагом является обеспечение одинаковых правил для всех случаев, в которых вводится или редактируется текст;

– разработка интерфейсов должна быть основана на постулате: *любые объекты, которые выглядят одинаково, одинаковы.*

Элементарные операции, порождаемые элементарными действиями:

– *Указание.* Пользователь может указать на то или иное содержание.

– *Выделение.* Пользователь может выделить какое-то содержание.

– *Активизация.* С помощью «клика» пользователь может активизировать содержание.

– *Модификация или использование (с помощью команд)*:

• *Генерация.* Модификация из «пустого» в «непустое».

• *Удаление.* Модификация из «непустого» в «пустое».

• *Перемещение.* Вставка содержания в одно место и одновременное его удаление из другого.

• *Трансформация.* Преобразование в другой тип данных.

• *Копирование.* Содержание может быть отправлено или получено от внешнего устройства или скопировано в другую область внутри системы.

Основные различия между программами. В основном *когнитивные различия между программами заключаются в способах представления выделенного содержания и того, как пользователь может с ним оперировать.*

Подсветка (highlighting) означает, что с помощью каких-либо средств отображенному на экране объекту придается заметное отличие. *Функция подсветки заключается в том, чтобы пользователь, пассивно наблюдая изображение на экране, мог определить, что некоторый объект получил от системы особый статус.* Подсветка единичного объекта при перемещении курсора без каких-то других действий со стороны пользователя (как, например, нажатие кнопки мыши) является указанием (indication)

Указание недостаточно используется в современных системах!

Активное использование указания в разработке интерфейса позволяет существенно сократить количество щелчков мышью в сравнении с современными интерфейсами.

Пример: *Активизация окна в современных операционных системах осуществляется щелчком. Если бы она была реализована через указание, системно для всей ОС, – это сократило бы число щелчков, необходимых для управления окном.*

Выделение (selecting) – это процесс, с помощью которого пользователь указывает, что один или несколько объектов имеют особый статус, который может быть воспринят системой. Как результат процесса получается выборка (selection). После того как выбор сделан, предыдущая выборка должна стать старой выборкой (old selection).

Недостатки способа выделения, использующегося в современных ПИ:

– команда для создания составных выборок является невидимой;

– при создании большой составной выборки легко допустить ошибку (например, если пользователь случайно отпустит клавишу <Shift> и щелкнет по следующему объекту, вся сложная выборка, которая была создана к этому моменту, будет потеряна);

– механизм используется как «переключатель»: один и тот же жест служит как для отмены выделения (если объект был уже выделен), так и для установки выделения (если объект был не выделен).

Возможное решение:

– решение первой проблемы: использование экранной подсказки;

– решение второй проблемы: ввод специальной команды, с помощью которой текущая выборка определяется как объединение старого и текущего выделения. Такая команда позволила бы пользователю сосредоточиться на создании выборки, не заботясь о том, что было выбрано до этого. Только после подтверждения текущего выделения она может быть добавлена к составной выборке;

– решение третьей проблемы: простым решением могло бы быть использование одной команды или квазирежима для добавления объекта к выборке, а другой команды или квазирежима – для удаления объекта из выборки.

Следствие из второй проблемы: в большинстве существующих сегодня систем команды Отменить (Undo) и Повторить (Redo) нельзя применить к процессу создания выборок. *Операторы Отменить и Повторить являются осно-*

вополагающими, и их функция настолько важна, что в будущих системах для них должна быть предусмотрена специальная клавиша.

Команды: команды не должны ограничиваться только меню, но могут быть частью вашего текста, или же, если это уместно, команда может быть представлена графическим объектом, а не словом или набором слов. *Важно также и то, что в этом случае пользователь может назначать команды самым простым способом – всего лишь набирая их с клавиатуры или рисуя.*

Пример (вариант 1): предположим, что имеется следующий текст: *Я возьму 3+4 яблока.*

При нажатии клавиши <*Вычислить*> он будет преобразован в следующий текст: *Я возьму 7 яблок.*

Пример (вариант 2): предположим, что имеется следующий текст: *Я хочу купить 3+4 рубашек вычислить.*

Выделим сумму 3+4 и затем слово вычислить, при этом сумма станет старой выборкой. При нажатии клавиши <*Command*> вызывается выбранная команда.

Принципы использования команд:

– синтаксис, который мы выбираем для написания команд, не должен исключать пробелы или символы новой строки;

– использование обычаев, которые не совпадают с традициями обычной речи, приводит к возникновению непонимания между пользователем и компьютером. Следует сделать так, чтобы машина подстраивалась под нас, а не подстраивать обычаи естественной речи под то, что проще решить с точки зрения программирования.

Побочный эффект – это следствие применения команды, при котором изменяется содержание или события, которые не находятся в локусе внимания пользователя. *Устранение побочных эффектов должно быть одной из целей для разработчика человекоориентированного интерфейса.*

Ошибки с удалением. Ввод символов замещает содержимое текущей *выборки*, что время от времени приводит к проблемам в тех случаях, когда новый материал неожиданным для пользователя образом удаляет текст, который он не собирался удалять. *Удаление текста должно проходить явным образом по желанию пользователя и не быть побочным эффектом другого действия.*

Имена файлов. *Необходимость давать имена файлам увеличивает ментальную нагрузку на пользователя. Назначение имени не делает ничего, кроме добавления к самому файлу еще нескольких символов.* Содержание текстового файла и есть его самое лучшее имя.

Виды организации интерфейса поиска:

– *поиск с разделителями (delimited search)* – наиболее распространенный вид поиска, который встречается в большинстве текстовых процессоров;

– *пошаговый поиск (incremental search)* – менее распространенный метод, известный пример которого можно увидеть в текстовом редакторе EMACS, работающем под операционной системой Unix, Linux.

Пошаговый поиск (incremental search). Когда пользователь вводит первый символ шаблона, система использует этот символ как полный шаблон и сразу же начинает поиск первого экземпляра этого символа в выбранном направлении. Если экземпляр этого первого символа обнаруживается до того, как введен следующий символ шаблона, то он выбирается, а курсор помещается сразу в конце выборки. Процесс повторяется по мере добавления символов к шаблону поиска.

Основной недостаток поиска с разделителями. Пользователь может ввести последовательность для поиска с опечаткой, но заметить ее слишком поздно, так как он уже нажал по привычке клавишу <*Return*>. Поэтому ему придется сидеть и ждать, пока закончится поиск, который, как уже заранее известно, не даст результата. *Большинство систем поиска являются непрерываемыми, и это является серьезной ошибкой разработчиков.*

Преимущества пошагового поиска:

– требует меньше времени;

– в нем имеется постоянная обратная связь во время введения символов шаблона, т.е. результаты поиска видны сразу.

Несмотря на то что разработчики и пользователи в основном признают, что пошаговый поиск более предпочтителен, почти все инструменты по разработке интерфейсов позволяют создавать средства поиска с разделителями и затрудняют или даже делают невозможным создание средства пошагового поиска.

***Общий принцип разработки человекоориентированных интерфейсов*:** *программа должна взаимодействовать с пользователем на основе наименьшей*

значимой единицы ввода. Пошаговый поиск является одним из примеров использования данного принципа.

Использование поиска в больших текстах. В больших по размеру текстах поиск может осуществляться по кругу (циклично) не только в локальном документе, но и далее, в автоматически расширяющихся областях вплоть до всего интернета.

Форма курсора (проблема первая). Одной из проблем стандартного текстового курсора является то, что пользователи пытаются поместить его точно между символами, целясь на небольшой горизонтальный объект, который по размеру меньше, чем требуемый, что в соответствии с законом Фитса приводит к временны́м затратам.

Выделение курсором (проблема вторая). Пользователь должен располагать курсор по-разному, в зависимости от того, какое действие он собирается совершить далее.

Возможное решение проблем с курсором – использование курсора, который визуально показывает как область вставки, так и символ или символы, которые могут быть удалены клавишей <*Backspace*>, может быть ценным улучшением интерфейсов, ориентированных на работу с текстами.

Дилемма вытеснения. Вы хотите выполнить некоторую задачу, которую могли бы выполнить в приложении *A*. Но вы находитесь в приложении *B*, в котором аналогичной команды нет. Вы должны знать о том, какое приложение в данный момент является активным, но вы не можете знать это наверняка, когда в локусе вашего внимания находится задача, которую вы пытаетесь выполнить. Поэтому иногда вы будете применять жесты, которые либо не будут давать результата, либо будут давать неправильный результат.

Известный способ решения дилеммы вытеснения. Наиболее распространенный способ заключается в том, чтобы каждое приложение снабдить всеми средствами, которые пользователю могут понадобиться *(OLE, OpenDoc Apple, NextWave HP).*

Основные недостатки:

– приложения разрастаются;

– приложения используют ограниченные по функциональности внешние средства.

Команды и трансформаторы. Программное обеспечение рассматривается человекоориентированным интерфейсом как набор команд, некоторые из которых являются трансформаторами, автоматически вызываемыми в тех случаях, когда тип данных, предусмотренных командой, не соответствует типу данных выбранного объекта. При этом может быть вызвано более одного трансформатора.

Пример работы команд и трансформаторов:

Предположим, что с помощью одного трансформатора производится конвертация из A в B, а с помощью другого – из B в C.

Если команда предусматривает обработку данных типа C, а данные выборки представляют собой тип A, то перед выполнением команды должны быть запущены два трансформатора.

Если необходимых трансформаторов не имеется, никаких изменений с выборкой не производится.

Пользователю при необходимости выдается специальное сообщение, а выборка остается без изменений.

Достоинство подхода: вместо прикладных программ разработчики будут поставлять наборы команд или трансформаторов, представляющие собой совокупность взаимосвязанных операций.

5. ПОНЯТИЕ МЕТАФОРЫ И МОДЕЛИ ПОЛЬЗОВАТЕЛЬСКОГО ИНТЕРФЕЙСА

Современный подход к проектированию интерфейсов рассматривает метафору как основную ментальную операцию, как способ познания, структурирования и объяснения мира. Метафора является предметом изучения нескольких научных дисциплин, например филологии, философии, науковедения. В последние десятилетия метафора стала предметом изучения специалистов по интерфейсу «человек–компьютер», визуальным коммуникациям и компьютерной визуализации.

Роль метафоры. Основная роль метафоры интерфейса заключается в том, что она способствует лучшему пониманию семантики взаимодействия, а также обеспечивает визуальное представление диалоговых объектов и определяет набор манипуляций пользователя с ними.

Виды метафор:

– *локальные проблемно ориентированные визуальные метафоры.* Для изучения параллельных вычислений и соответствующих операционных систем, а также метафоры, основанные на использовании бытовых и общеизвестных технических понятий.

– *глобальные метафоры* (метафора рабочего стола) и др.

Известные компьютерные метафоры:

– *метафора рабочего стола.* Стала визуальной метафорой;

– *метафора рабочей комнаты.* Стала визуальной метафорой, но не прижилась в ПИ ОС;

– *метафора всемирной паутины.* Не стала визуальной метафорой.

Недостатки метафор. Перенос значения, который поддерживает пространственные визуальные метафоры посредством сходства или аналогий с ситуациями реального мира, может быть как позитивным, так и негативным, когда на метафорическое значение переносятся ограничения реальных ситуаций. Часто при использовании метафоры проявляются «метафорические артефакты», т.е. на компьютерную модель переносятся отдельные свойства объектов метафоры, не существующие в исходной постановке.

Ментальная модель. Ментальная, или концептуальная, модель – лишь внутреннее отображение того, как пользователь понимает и взаимодействует с

системой. Это отображение (в основном) физической системы или компьютерного программного обеспечения, в котором изложена вероятная последовательность действий при выполнении операций ввода и вывода. В основе ментальной модели лежат все взаимоотношения между пользователями и их компьютерами, поэтому она является фундаментом для выработки принципов и правил пользовательского интерфейса.

Модели интерфейса:

– *модель пользователя* – опыт взаимодействия в реальном мире. Включает задачи, процессы, инструменты, результаты. Представляет собой данные от разных групп людей, которые имеют разные персональные, профессиональные и компьютерные привычки и наклонности, описывающие решение задач с помощью ПО, для которого разрабатывается интерфейс;

– *модель проектировщика.* Включает принципы и методы проектирования ПИ. Представляет собой объединение идей и пожеланий пользователя, соединённое с навыками проектировщика и материалами, необходимыми для программиста; является проектом, удовлетворяющим нужды пользователя;

– *модель программиста.* Включает платформу ОС, оболочку, инструменты разработки, принципы и методы разработки. Может быть формально и недвусмысленно описана и может быть представлена в определенном виде функциональной спецификацией программного продукта.

6. ПРОЕКТИРОВАНИЕ ПОЛЬЗОВАТЕЛЬСКОГО ИНТЕРФЕЙСА

Существует четыре этапа разработки пользовательского интерфейса:

– сбор и анализ информации от пользователя;

– разработка ПИ;

– построение ПИ;

– подтверждение качества созданного ПИ.

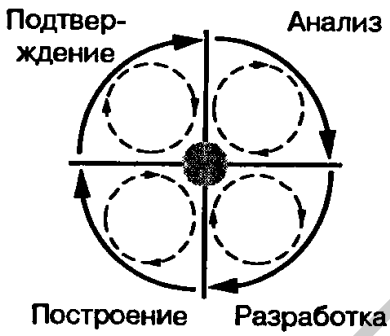

Рис. 6.1. Этапы разработки ПИ

Этап 1. Сбор и анализ информации от пользователя.

Шаг 1. Определение профиля пользователя – что представляет собой наш пользователь?

Шаг 2. Анализ стоящих перед ним задач – определение того, чего хотят пользователи и каким образом они собираются решать свои задачи.

Шаг 3. Сбор требований, предъявляемых клиентами, – определение, какую пользу с точки зрения пользователя принесет им предлагаемый продукт или интерфейс?

Шаг 4. Анализ рабочей среды пользователей – где пользователи решают стоящие перед ними задачи?

Шаг 5. Соответствие требований пользователей стоящим перед ним задачам – проверка стоящих перед пользователям задач на реалистичность.

Определение профиля пользователя позволяет получить данные о возрасте, образовании и предпочтениях пользователя и другую необходимую информацию.

Анализ стоящих перед пользователем задач. Независимо от метода анализа задач необходимо получить ответы на следующие вопросы:

1. Какие задачи решают пользователи?
2. Какие задачи являются наиболее важными?
3. Какие шаги предпринимаются для решения задач?
4. Какие цели преследуют пользователи при решении тех или иных задач?

5. Какой информацией необходимо располагать для выполнения задач?

6. Какой инструментарий (компьютеры и т.д.) используются для решения задач?

7. Каков ожидаемый итог от решения задачи?

8. Каким образом пользователи выполняют свою работу (вручную, на компьютере, по телефону и т.д.)?

9. Каким образом они взаимодействуют с другими лицами при решении задач?

10. Каким образом задачи учитываются в общем бизнес-процессе?

11. Как часто пользователям приходится решать задачи?

12. Каким образом компьютер или другая компьютерная техника помогает пользователям в решении задач?

Сбор требований, предъявляемых клиентами:

– какие основные технологии требуются пользователям?

– сколько пользователи и менеджеры готовы заплатить за продукт?

– кто устанавливает продукт?

– кто будет сопровождать продукт, когда он будет установлен?

Анализ рабочей среды пользователей. Информация собирается в отношении:

– физической стороны рабочей среды (освещение, шум, рабочее пространство, температура, наличие компьютеров, телефонов, количество персонала и т.д.);

– места работы пользователя и степень его мобильности (офис, квартира, стационарно, с передвижениями и т.д.);

– вопросов эргономики, условий труда (задействуются ли зрение, слух, работа ведётся стоя/сидя, на клавиатуре и т.д.);

– особых запросов (уровень подготовки, физическое состояние, интерес к познавательному процессу, особенности речи и возможные недостатки);

– интернационализации и других культурологических условий (перевод, цвета, иконки, текст, сообщения и т.д.).

На этапе ***соответствия требований стоящим перед пользователем задачам*** проверяется соразмерность требований пользователя решаемым задачам, то есть реалистичность требований, предлагается оптимальных вариант требований с учетом соразмерности требований задачам.

Этап 2. Разработка пользовательского интерфейса.

Шаг 1. Определение цели с точки зрения применения продукта.

Шаг 2. Разработка задач и сценария действий пользователя.

Шаг 3. Определение целей и операций интерфейса.

Шаг 4. Определение иконок объектов и визуального представления.

Шаг 5. Разработка меню объектов и окон.

Шаг 6. Оптимизация визуальной разработки.

Шаг 1. Определение цели с точки зрения применения продукта. Определить, какова польза продукта. Наиболее общие области для формулировки целей:

– пригодность;

– эффективность;

– легкость в освоении;

– оценка пользователями качества продукта.

Шаг 2. Разработка задач и сценария действий пользователя. Как правило, сценарий является последовательным перечислением действий, которые должен выполнить пользователь для решения одной из задач, стоящих перед ним. *Чем больше сценариев – тем лучше! Сценарии должны быть понятны всем участникам разработки!*

Шаг 3. Определение целей и операций интерфейса. На этом этапе необходимо:

– выделить объекты, данные и действия из задач, стоящих перед пользователем;

– посмотреть и уточнить список объектов и действий совместно с пользователями;

– начертить диаграмму взаимодействий между объектами;

– заполнить матрицу прямого манипулирования объектами.

Таблица 6.1

Схема матрицы прямого манипулирования

Исходный объект	*Конечный объект*				
	Объект1	*Объект2*	*Объект3*	*…*	*ОбъектN*
Объект1		*Действие 1*			
Объект2					
Объект3	*Действие 2*				*Действие M*
…					
ОбъектN			*Действие 3*		

Шаг 4. Определение иконок объектов и визуального представления. На этом этапе работает специалист по графике. Визуальные формы должны быть легко узнаваемы и по возможности однозначно соответствовать выделенным объектам. *Не забываем об унификации внешнего представления элементов!*

Шаг 5. Разработка меню объектов и окон:

– какие действия свойственны каждому объекту и типу?

– что содержится во всплывающих меню?

– каким окнам требуется всплывающее меню?

Шаг 6. Оптимизация визуальной разработки. При первой разработке схем окон удобнее пользоваться бумагой и карандашом. Когда вид окна определен, можно перейти к инструментам прототипирования и проектирования.

Этап 3. Построение пользовательского интерфейса.

Необходимо следовать трём золотым правилам:

1. Прототипируйте на ранних стадиях и не забывайте итерационные процесс разработки.

2. Создавайте различные альтернативные варианты.

3. Будьте готовы выбросить код прототипа.

Этап 4. Подтверждение качества созданного пользовательского интерфейса.

Не позволяйте разработчику тестировать программный продукт!

Приглашаем к тестированию потенциальных пользователей!

Используем психологов, не относящихся к программному продукту!

7. МОДЕЛЕОРИЕНТИРОВАННЫЙ ПОДХОД К РАЗРАБОТКЕ ИНТЕРФЕЙСОВ

7.1. Средства разработки интерфейса, основанные на моделях

Термин «средства разработки интерфейса, основанные на моделях» относится к средствам создания интерфейса, которые предлагают разработчикам специфицировать не сам интерфейс, а его модель. *К таким средствам относятся системы автоматической генерации интерфейса, генераторы систем помощи для приложений, средства оценки интерфейса и др.*

Основная цель таких средств – автоматическая генерация пользовательского интерфейса на основе декларативных описаний.

Автоматическая генерация пользовательского интерфейса снижает стоимость разработки; возрастает модульность спецификации (за счет различных моделей), становятся доступными дополнительные возможности, например, генерация контекстно-зависимой помощи.

Понятие модели в рамках моделеориентированного подхода. Модель является основным информационным компонентом любой МОС (моделеориентированной системы). При этом каждая такая система имеет свой набор моделей, необходимый для проектирования интерфейса, а также язык моделирования. *Модель определяется как декларативное описание, не содержащее процедурного кода и состоящее из описательных предложений высокого уровня абстракции.*

7.2. Три уровня абстракции модели интерфейса

Первый уровень абстракции. *Модель задачи* представляет задачи, которые пользователю необходимо решить с помощью приложения. *Модель предметной области* представляет данные и операции, которые поддерживает приложение.

Второй уровень абстракции. Абстрактная спецификация пользовательского интерфейса, которая описывает структуру и содержание интерфейса в терминах трех абстракций:

– *объектов абстрактного взаимодействия*, представляющих собой низкоуровневые интерфейсные задачи (например, выбор элемента из множества);

– *информационных элементов*, представляющих данные, которые должны быть показаны пользователю (например, множество объектов и атрибутов из предметной области);

– *единиц представления* – абстракций окон, определяющих элементы, которые должны быть показаны в рамках одной единицы представления.

Третий уровень абстракции. Конкретная спецификация интерфейса, определяющая интерфейс в терминах набора примитивных инструментов, таких как окна, кнопки, меню, переключатели и т.д. Модель интерфейса позволяет ответить на следующие вопросы, касающиеся конкретного интерфейса:

– Кто пользователи интерфейса?

– Какие задачи они собираются решить с помощью интерфейса?

– Какие компоненты интерфейс предоставит его конкретному пользователю?

– Какие функции реализует пользовательский интерфейс?

Различные МОС имеют свои наборы моделей. В *Mastermind* необходимо построить три основные модели – представления, диалога и взаимодействия. В *Mecano* требуется построить модели пользователя, задачи, предметной области, представления, диалога и дизайна. *ITS* имеет три уровня или модели пользовательского интерфейса – модель предметной области, называемая фондом данных, спецификация содержимого (абстрактная спецификация интерфейса, которая содержит списки, формы, элементы выбора, информационные блоки и другую информацию, которая будет представлена пользователю), спецификация стиля (определяет внешний вид и способ взаимодействия). *Tellach* использует три модели, которые описывают различные аспекты пользовательского интерфейса: модель предметной области, модель задачи и модель представления. Также имеется модель отношений для описания связей между моделями.

Модель предметной области описывает структуру и атрибуты информации, предоставляемой приложением, – классы, наследования классов, атрибуты каждого класса, а также отношения между объектами. *Модель предметной области* определяется как набор объектов предметной области, организованный в классы, к которым пользователь может обращаться через интерфейс. *Различные МОС используют свои модели данных для представления понятий предметной области*.

Выразительные средства диалога в модели интерфейса. В большинстве МОС выразительные средства диалога описываются в так называемых моделях

представления. *Модель представления* определяет интерактивные объекты или заготовки (widgets), которые появляются на экране в различных состояниях диалога. В общем случае модель представления состоит из иерархической декомпозиции возможных экранов, состоящих из групп интерактивных объектов. *Методы реализации моделей представления отличаются в различных МОС.*

Сценарии диалога в модели интерфейса. Модели диалога и представления близки по смыслу друг к другу и в большинстве МОС объединены в одно целое, а методы их реализации отличаются в различных МОС. В ряде МОС эти модели отделены друг от друга. Сценарий диалога в таких системах задается чаще всего в модели диалога (взаимодействия). *Модель диалога* описывает диалог «человек–компьютер». Эта модель специфицирует:

– когда конечный пользователь может запустить различные функции через так называемые запускающие механизмы (кнопки, команды и т.д.) и устройства мультимедиа (речевой ввод, сенсорные устройства и т.д.);

– когда конечный пользователь может выбрать или специфицировать ввод;

– когда компьютер может запросить информацию у конечного пользователя.

Прикладная программа в модели интерфейса. Связь между прикладной программой и интерфейсом отображается в большинстве МОС в модели задач. *Модель задач* описывает задачи, которые необходимо выполнить пользователю. Модель задач обычно иерархически разбивает каждую задачу на подзадачи до тех пор, пока конечные задачи не представляют собой операции, выполняемые приложением (например, and, or, parallel и др.). Каждая задача содержит цель, условия выполнения, описание результатов ее выполнения, требования к информации (какая информация необходима для выполнения задач) и схему разбиения на подзадачи. *Схема задачи* определяет комбинацию пользовательских задач, интерфейсных задач и задач приложения. Задача моделируется в терминах объектов, называемых задачами. *Модель задачи пользователя* – это множество иерархически организованных задач пользователя. *Задача пользователя* – это определение деятельности, которую пользователь желает выполнить. Задача имеет конечную цель (логическое выражение) и может быть разбита на несколько подзадач. Подзадачи выполняются согласно заданному порядку.

Связи между составными частями модели интерфейса. Модели интерфейса описывают всю информацию, необходимую для разработки интерфейса. Однако остается много неявной информации или связей между моделями, ко-

торые очень важны, особенно в случае изменения моделей. *Проблема связывания* возникает в любой многомодельной системе между любой парой связываемых моделей. Связывание (binding) бывает затруднительным потому, что модели могут иметь различные принципы и механизмы формирования своих элементов (составляющих модели), что не позволяет разработать какой-либо универсальный метод связывания.

7.3. Проектирование интерфейса с помощью МОС

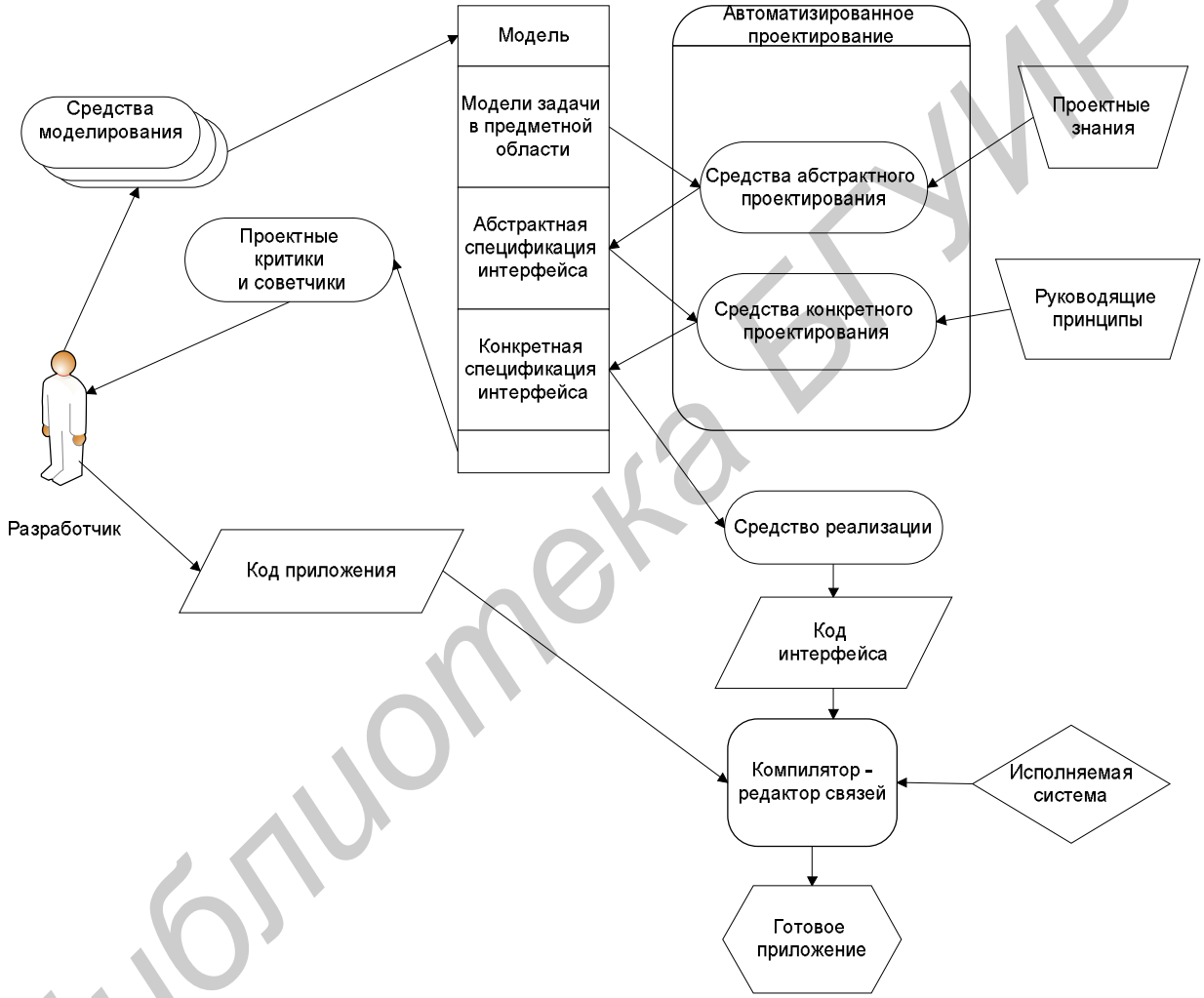

Рис. 7.1. Архитектура моделеориентированных средств для разработки пользовательского интерфейса

Как правило, разработка пользовательского интерфейса при моделеориентированном подходе опирается на средства с архитектурой подобной архитектуре, приведенной на рис. 7.1. Сначала с помощью специализированных инструментальных средств, предлагающих ряд функций и уровней автоматизации, разрабатываются модели интерфейса. Затем модель интерфейса может быть

преобразована в выполнимую спецификацию (составленную по типу программы с использованием языка спецификаций).

Три вида средств реализации ПИ:

– генераторы кода генерируют исходный код на языке программирования;

– генераторы СУПИ генерируют файл, который может быть прочитан существующими СУПИ или средствами построения интерфейса;

– интерпретаторы интерпретируют модель во время исполнения программы.

Многие МОС предоставляют средства реализации, использующие модель для генерации пользовательского интерфейса. В настоящее время известны следующие МОС: Mastermind, Mecano, ITS, Tellach, Genius, Humanoid, UIDE, MOBI-D.

Проектирование интерфейса в MOBI-D проходит в рамках следующих этапов:

– *построение модели задач пользователя.* Интерактивные инструменты MOBI-D позволяют пользователю вводить текстовое описание задачи. Инструмент выявляет ключевые термины, такие, как объекты (существительные) и действия (глаголы) для построения моделей предметной области и модели задач соответственно;

– *интеграция моделей.* Затем модели (с помощью инструментов) интегрируются так, чтобы объекты предметной области были связаны с задачами пользователя, для которых они предназначены. Проектирование и представление диалога являются параллельными действиями.

– *генерация интерфейса.* Инструменты принятия решений исследуют модель задачи и предметной области для формирования рекомендаций для элементов управления и взаимодействия. На основе этих действий генерируется готовый интерфейс.

Построение пользовательского интерфейса средствами Mastermind проходит в рамках следующих этапов:

– *конструирование компонент. М*одели представления, оболочки приложения и модели диалога. *Разработка компонента, ответственного за связывание*. Первый шаг – связывание (с помощью общего именования прикладных методов) представления и поведения системы (ее диалог). Второй шаг – разработка модели контекста. С ее помощью проектировщик определяет, какая дополнительная информация должна быть разделена между компонентами, какие

данные программной системы отражены в представлении, или какое действие может выполнить пользователь;

– *использование генераторов кода.* Каждый генератор кода формирует исходные файлы на C++, которые затем компилируются и связываются с рядом динамических библиотек;

– *соотнесение моделируемых образцов с элементами других представлений модели.* Образцы моделей представляются в текстовом формате MTF (Mastermind textual format), который удобен для редактирования пользователем. Эти файлы используются и во время выполнения приложения (например, чтобы поддерживать контекстно-зависимую справку).

8. АДАПТИВНЫЕ ПОЛЬЗОВАТЕЛЬСКИЕ ИНТЕРФЕЙСЫ

8.1. Основа адаптивного подхода

Главный недостаток WIMP-интерфейсов **–** невозможность использования таких каналов взаимодействия, как речь, слух и прикосновения. Хотя большое количество наших нейронов находится в «визуальной» части коры головного мозга, что позволяет зрению быть информационным каналом с самой высокой пропускной способностью, все равно без речи, слуха и прикосновений общение с физическим миром не может быть полноценным. В основе адаптивного подхода лежит создание гибкой структуры диалога, легко адаптирующейся к различным пользователям, когда структура взаимодействия «человек–машина» изменяется непосредственно в процессе общения. Гибкий адаптивный подход подразумевает автоматическую настройку программного окружения вычислительной системы в соответствии с рядом особенностей конкретного пользователя (таких, как уровень знакомства пользователя с системой, его профессиональная подготовленность, ряд психологических особенностей и т.д.).

Адаптивный интерфейс – интерфейс, основанный на гибкой структуре диалога, который в любой текущий момент времени обеспечивает конкретного пользователя наиболее удобным типом интерактивного взаимодействия из числа предусмотренных в системе.

Основная цель адаптации интерфейса – предоставление пользователю только тех функциональных возможностей, в которых он нуждается при выполнении конкретной задачи. При таком подходе пользовательский интерфейс создает меньше проблем, облегчается доступ ко всем необходимым функциональным возможностям, увеличивается скорость выполнения задач. Более того, адаптивность интерфейса подразумевает, что пользователю могут быть предоставлены возможности, которые могут не соответствовать оптимальной конфигурации в каждой конкретной рабочей ситуации, но которые позволят, в особенности неопытным пользователям, наиболее легко взаимодействовать с системой. Адаптивные интерфейсы расширяют взаимодействие между пользователем и вычислительной системой за счет:

– увеличения диапазона способов ввода и вывода;

– обогащения грамматики используемой информации;

– попытки кооперации с пользователем в достижении целей задачи.

Выделяют ***три стадии адаптационного процесса*** (рис. 8.1)***:***

– сбор информации о пользователе;

–обработка данных и построение или обновление модели пользователя;

– применение модели пользователя для достижения эффекта адаптации.

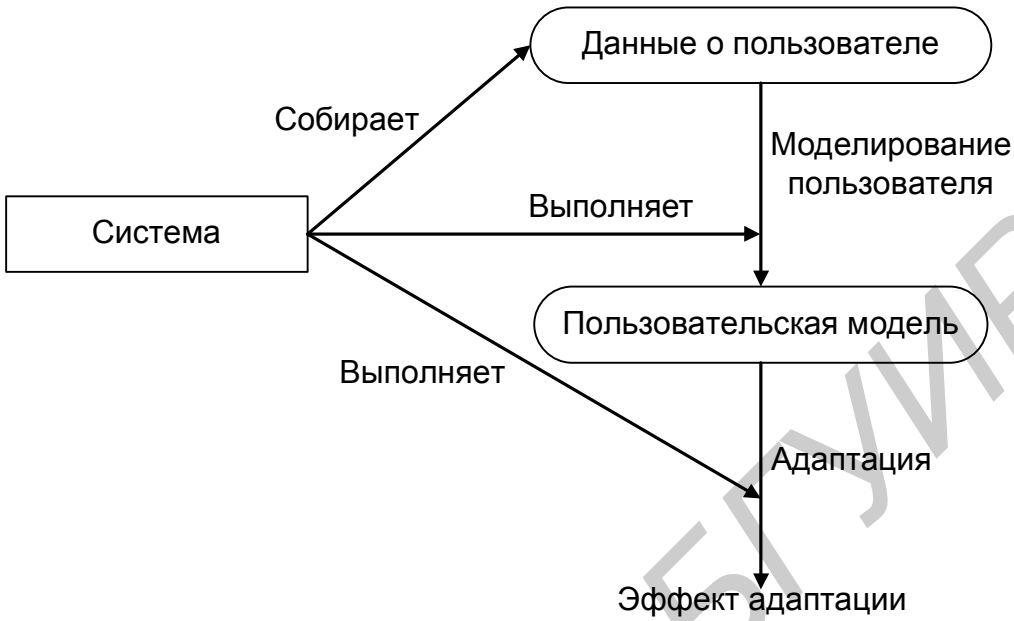

Рис. 8.1. Классический цикл «Моделирование пользователя – адаптация» в адаптивных системах

Можно назвать следующие ***отличия адаптивного интерфейса от обычного:***

– интерфейсами адаптивных систем обеспечивается разумная мера гибкости – избыточная гибкость может привести к сложной системе, которая превратит преимущество в существенный недостаток;

– подбор диалоговой структуры взаимодействия «человек–машина» для каждого конкретного пользователя осуществляется в контексте предметной области, содержащей правила интерфейсной настройки;

– адаптивному интерфейсу доступен расширенный диапазон входных сообщений и представлений выходной информации;

– предусматривается возможность устранения двусмысленностей или анализ нераспознаваемых входных сообщений с помощью взаимодействия с пользователем;

– интерфейс может обладать информацией о внимании пользователя. В большинстве адаптивных систем осуществляется простейший анализ ошибок, допущенных пользователем; имеются возможности автоматического исправления простейших опечаток;

– интерфейс должен уметь определять цели, преследуемые пользователем при вводе информации. Предусматривается возможность ведения переговоров в случае обнаружения двусмысленностей или нераспознаваемых входных сообщений;

– система, основанная на концепции адаптивности, позволяет человеку преодолевать слабости и максимально полно использовать свои способности для достижения тех целей, ради которых он воспользовался услугами компьютера.

Для улучшения взаимодействия системы с пользователем требуется, чтобы она обладала моделью пользователя!

Выделяются *два основных направления в разработке адаптивных систем* – построение программных приложений, которые предоставляют пользователю возможности настройки интерфейса (adaptable systems), и системы, которые производят эту настройку автоматически (adaptive systems).

Две методики организации процесса интерфейсной адаптации:

– *индивидуальная.* Говорят, что адаптивный интерфейс системы спроектирован на основе *индивидуальной методики* (Individual Methodics) или что данная система поддерживает *индивидуальную интерфейсную адаптацию*, если каждому пользователю вычислительной системы ставится в соответствие своя, отличная от других (за редким исключением) модель пользователя;

– *стереотипная.* Если вычислительной системой анализируется принадлежность каждого реального пользователя к определенной допустимой модели (классу), то говорят, что ее адаптивный интерфейс спроектирован на основе стереотипной методики (Stereotyping Methodics) или что данная система поддерживает стереотипную интерфейсную адаптацию.

Важнейшее отличие индивидуального и стереотипного видов интерфейсной адаптации. В случае *индивидуальной методики* для каждой сгенерированной модели пользователя формируется свой персонифицированный диалог на основании правил предметной области, в то время как при *стереотипной адаптации* возможен вариант, когда в соответствии с заданной классификацией разработчик заранее проектирует свой особый интерфейс для каждого класса системных пользователей.

По времени осуществления адаптации можно выделить системы, производящие интерфейсную адаптацию для каждого конкретного пользователя один раз – при первом сеансе взаимодействия с ним, и системы, в которых

адаптация пользовательского интерфейса реализуется на основании данных, получаемых в ходе каждого сеанса работы. ***Наиболее перспективный принцип – проектирование программ*** с *динамической интерфейсной адаптацией*, осуществляющейся автоматически во время взаимодействия с пользователем на основании данных, полученных как в ходе первоначального тестирования, так и в ходе анализа каждого сеанса работы.

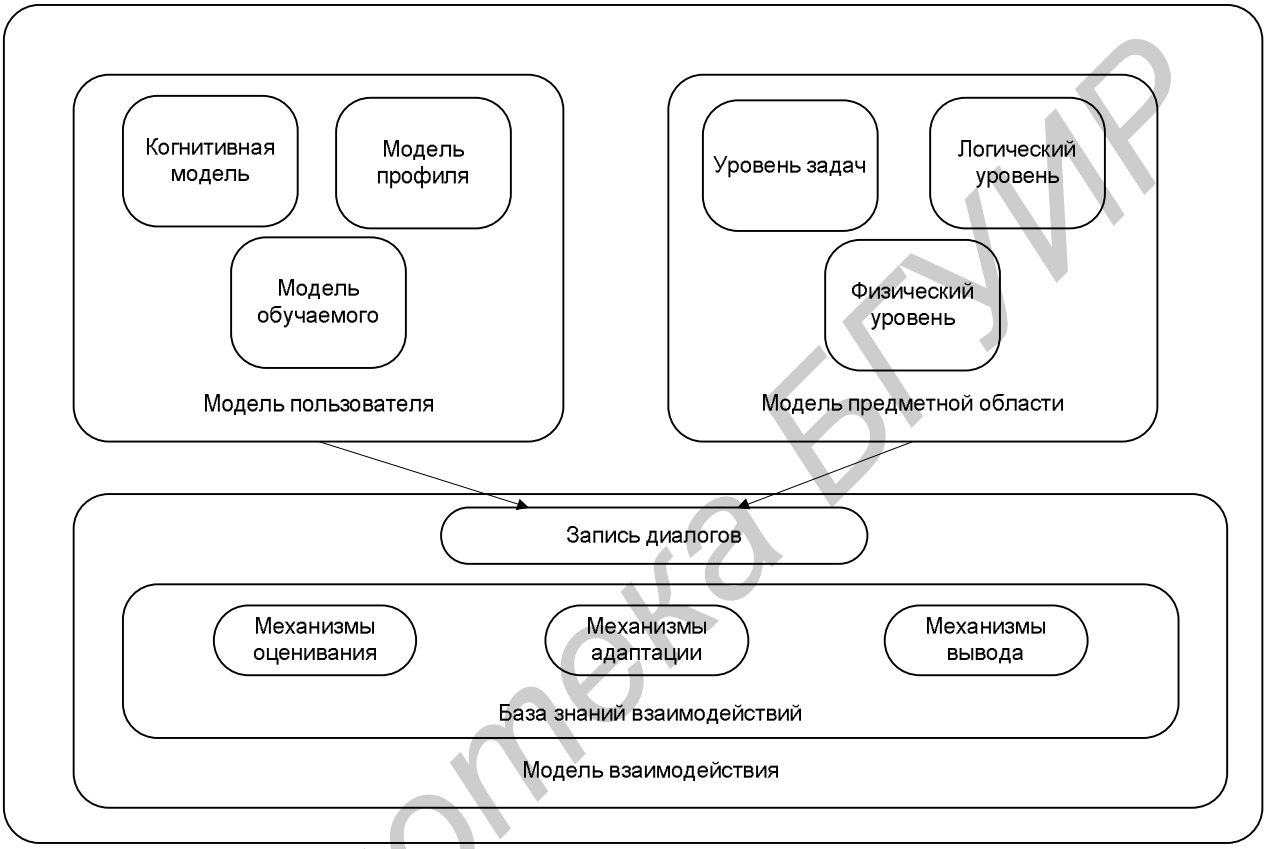

Рис. 8.2. Общая архитектура адаптивной системы

Основные направления работ в области адаптивных интерфейсов:

– построение программных систем с автоматической динамической адаптацией;

– адаптация на основании биометрических характеристик;

– подход на основе технологии «чутких» компьютеров;

– адаптация интерфейсов на основании анализа речи пользователей.

Адаптация на основе биометрических характеристик. Для управления компьютером используется выражение лица человека, направление его взгляда, размер зрачка и другие признаки. Для идентификации пользователя используется рисунок радужной оболочки его глаз, отпечатки пальцев и другая уникальная информация. Изображения считываются с цифровой видеокамеры, а

затем с помощью специальных программ распознавания образов из этого изображения выделяются команды.

Подход на основе технологии «чутких» компьютеров. Концепция создания чутких (отзывчивых) компьютеров состоит в том, что машине следует фиксировать изменения в настроении пользователя и соответственно на них реагировать. При оценке настроения партнера важной (как подчеркивается в зарубежных работах) является проблема фрустрации (user frustration) – реакция компьютера как на короткие эмоциональные всплески – вспышки гнева пользователя, так и на долговременное негативное состояние, когда «всё не ладится». Для оценки настроения партнёра, а также для уточнения его психического типа следует использовать опыт исследования влияния сознания человека, его психоэмоционального состояния на специальные приборные системы со специальными датчиками.

Адаптация интерфейсов на основании анализа речи пользователей. Появление и развитие мультимедийных технологий, исследование в области распознавания и анализа речи – все это способствует дальнейшему развитию данного подхода. Анализ речи позволит выявить новые характеристики, которые существенно могут повлиять на интерфейсную адаптацию.

8.2. Модель пользователя

Под моделью пользователя понимают представление о пользователе, которое формируется системой либо на основании заранее собранной информации, либо в ходе наблюдений за его взаимодействием с программным приложением. Модель пользователя в области HCI имеет более широкую интерпретацию. Она рассматривается как некоторая модель, отражающая психологические особенности пользователя. Такие модели позволяют предугадать и объяснить взаимодействия системы с человеком. ***Формирование модели пользователя****:* формирование модели пользователя в каждой адаптивной системе организовано по-своему. Традиционно для выполнения интерфейсной настройки формирование модели (профиля, портрета) пользователя и (или) серии рекомендаций осуществляется в специальной компоненте, предназначенной для тестирования пользователей. Тестирование обычно состоит из трех этапов – генерации, непосредственного тестирования и интерпретации результатов.

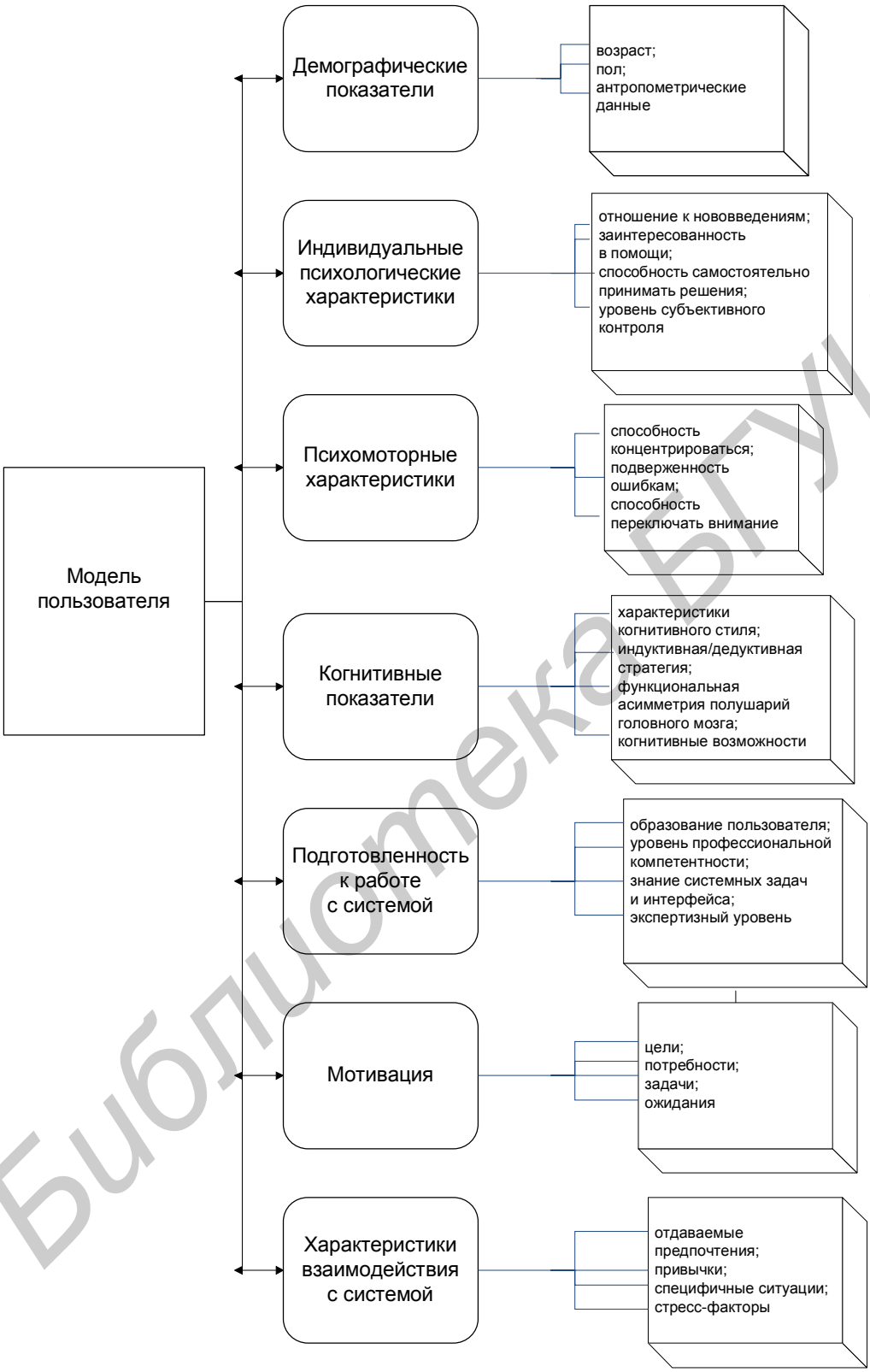

Рис. 8.3. Обобщенная структура модели пользователя

Основные способы организации процесса тестирования. При разработке адаптивных программных систем используются следующие виды тестирования:

– *текстово-опросное тестирование* во многом напоминает традиционное анкетирование или интервьюирование. Оно подразумевает наличие некоторого опросника, оформленного соответствующим образом;

– *игровое тестирование.* Под *игровым тестированием* подразумевается такой вариант проведения экспериментов, в котором в качестве основных форм представления информации используются графика, видео, анимация, звук или элементы технологии мультимедиа. Игровое тестирование позволяет моделировать различные ситуации, а портрет испытуемого в ходе данного тестирования формируется путем скрытого наблюдения за тем, как он решает возникающие проблемы или выполняет системные задания;

– *тестирование «в ходе слежения за пользователем»* по характеру своего построения сходно с игровым тестированием. Разница заключается в том, что в данном случае изучается поведение пользователя не в процессе моделируемых игровых ситуаций, а непосредственно в ходе эксплуатации диалоговой системы. С помощью данной компоненты тестирования система как бы «следит» за пользователем и подстраивает свое программное окружение в соответствии с изменившимися показателями, например, такими, как уровень экспертизы, мотивация или сложившиеся привычки.

8.3. Применение модели пользователя для интерфейсной адаптации

В качестве критериев интерфейсной адаптации обычно используют те характеристики пользователей, которые оказывают наибольшее влияние на характер интерактивного взаимодействия, а также непосредственно на сам процесс решения задач в рамках разрабатываемой системы.

Задача исследователей в области HCI – составить методику разработки систем, которые будут наиболее полно отвечать требованиям людей, соответствовать их психологическим и другим особенностям.

Для применения модели пользователя необходимо установить взаимосвязь между ее параметрами и компонентами интерфейса!

Демографические характеристики :

– *возраст*. Используется для идентификации пользователей довольно часто. Оказывает влияние на такие параметры интерфейса, как формат представления информации, уровень справочной информации, а также на большинство оформительских параметров;

– *пол*. Характеристика обычно применяется в том случае, когда большинство критериев представлено психологическими или психомоторными качествами человека, так как именно пол определяет разницу в психологическом складе мужчины и женщины.

Индивидуально-психологические характеристики :

– *обучаемость*. Определяет отношение человека к нововведениям, способность менять установки и точку зрения в соответствии с изменяющейся ситуацией;

– *конформизм.* Характеризует степень зависимости человека от окружающего мира и, в частности, от других людей. Низкая оценка данного параметра свидетельствует о том, что человек зависим от группы, следует за общественным мнением, ориентируется на одобрение извне;

– *уровень субъективного контроля.* Характеризует способность пользователя самостоятельно принимать решения. Возможны два полярных типа такой локализации контроля: экстернальный и интернальный. Людям с экстернальным локусом в большей степени присуще конформное и уступчивое поведение. Интерналы не склонны подчиняться давлению других. Люди с интернальным локусом контроля предпочитают работать в одиночестве. Экстерналы же, напротив, активно ищут вспомогательную информацию, поскольку решить проблему без посторонней помощи, не имея достаточного опыта, они практически не в состоянии. Метод проб и ошибок они предпочитают не использовать.

Психомоторные характеристики :

– *внимание*. Характеризует способность пользователя концентрироваться и переключаться в процессе решения конкретной задачи;

– *подверженность ошибкам.* Определяет степень развитости самоконтроля. Человеку с развитым самоконтролем свойственна точность выполнения рекомендаций и ограничений системы. При высоких оценках наблюдается недисциплинированность, человек не обеспокоен выполнением системных требований.

Когнитивные характеристики:

– *типы логического мышления.* Любой человек в большей или меньшей степени тяготеет к дедуктивному или индуктивному типу. Данную характеристику необходимо учитывать при организации справочной информации. Объяснение должно быть построено таким образом, чтобы информация, содержащаяся в нем, была правильно интерпретирована пользователем, независимо от того, какой когнитивной стратегии он придерживается;

– *вербальный – невербальный интеллект*. Отражает асимметрию человеческого мышления, которая проявляется в том, что люди по-разному воспринимают информацию в зависимости от формы ее представления , что обусловлено неодинаковым развитием полушарий головного мозга. Людям с более развитым левым полушарием в большей мере свойственно левостороннее логическое мышление, а людям с более развитым правым полушарием – соответственно правостороннее образное мышление.

Характеристики общей подготовленности :

– *опыт работы в предметной области.* Позволит обеспечить каждого пользователя механизмом навигации, соответствующим его знаниям и интересам;

– *образование пользователя.* Назначение этой характеристики – получение сведений о том, с какими предметными дисциплинами, необходимыми для эксплуатации системы, знаком пользователь;

– *экспертизный уровень.* Присваивается пользователю в определенный момент времени и обусловлен наличием у последнего соответствующих навыков работы с системой и непосредственно с компьютером;

– *перцептуальные ошибки.* Результат неполных перцептуальных (визуальных) очередей. Большое количество ошибок данного рода указывает на то, что пользователь еще недостаточно изучил систему поддержки данного экспертизного уровня – подсказки, справочные инструкции и т.д.;

– *когнитивные ошибки.* Результат недостаточности опыта для решения задач или свидетельство переутомления человека;

– *моторные ошибки.* Чаще называемые опечатками и составляющие почти треть всех ошибок, совершаемых пользователями, есть следствие нарушения у пользователя координации в результате утомления рук и глаз.

Характеристики взаимодействия с системой:

– *текущие задачи.* Подразумевает задачи, решаемые пользователем в процессе эксплуатации системы;

– *характеристика, связанная с получением базовых знаний о предметной области*. Будет оказывать влияние на такие параметры интерфейса, как набор доступных рабочих процессов, механизм навигации, состав меню.

Способы описания модели пользователя. Используя основные понятия теории множеств, модель пользователя можно описать следующим образом:

$$UM = \bigcup_{i=1}^{m} A_i ,$$

где m – число факторов, составляющих модель пользователя;

A_i – множества, задающие i-й фактор, учтенный в модели пользователя.

Модель пользователя также может быть изображена в виде фреймоподобной концептуальной структуры. Например:

UM	A_i
Возраст	*молодой* | *средних лет* | *пожилой*
Пол	*мужской* | *женский*
УСК	*интернал* | *экстернал*
Внимание	*низкое* | *среднее* | *высокое*
Функциональная асимметрия мозга	*левополушарность* | *баланс* | *правополушарность*
Тип темперамента	*сангвиник* | *холерик* | *флегматик* | *меланхолик*
Уровень образования	*начальное* | *среднее* | *высшее*
Навыки работы с компьютером и в сети Интернет	*новичок* | *пользователь* | *продвинутый пользователь*

ЛИТЕРАТУРА

1. Мандел, Т. Дизайн интерфейсов : пер. с англ. / Т. Мандел. – М. : ДМК Пресс, 2005. – 416 с.

2. Интеллектуальные обучающие системы и виртуальные учебные организации : монография/ В. В. Голенков и др.; под ред. В. В. Голенкова, В. Б. Тарасова. – Минск : БГУИР, 2001. – 488 с.

3. Грибова, В. В. Концепция разработки пользовательского интерфейса на основе онтологий В 2 ч. Ч. 1. Инструментарий для разработки пользовательского интерфейса (обзор литературы). Основная идея подхода /В. В. Грибова, А. С. Клещев. – Владивосток : ИАПУ ДВО РАН, 2003. – 24 с.

4. Грибова, В. В. Концепция разработки пользовательского интерфейса на основе онтологий. В 2 ч. Ч. 2. Модель пользовательского интерфейса / В. В. Грибова, А. С. Клещев. – Владивосток : ИАПУ ДВО РАН, 2003. – 51 с.

5. Раскин, Дж. Интерфейс : новые направления в развитии компьютерных систем / Дж. Раскин; пер. с англ. – СПб. : Символ-Плюс, 2004. – 272 с.

6. Поспелов, Д. А. Интеллектуальные интерфейсы для ЭВМ новых поколений /Д. А. Поспелов // Электронная вычислительная техника. Сб. ст. Вып. 3. – М. : Радио и связь, 1989. – С. 4–20.

7. Авербух, В. Л. К теории компьютерной визуализации / В. Л. Авербух // Вычислительные технологии. – 2005. – Т. 10. – № 4. – С. 31.

Св. план 2012, поз. 20

Учебное издание

Голенков Владимир Васильевич
Гулякина Наталья Анатольевна
Колб Дмитрий Григорьевич

ИНТЕЛЛЕКТУАЛЬНЫЙ ПОЛЬЗОВАТЕЛЬСКИЙ ИНТЕРФЕЙС

УЧЕБНО-МЕТОДИЧЕСКОЕ ПОСОБИЕ

Редакторы *Т. П. Андрейченко, Г. С. Корбут*
Корректор *Е. Н. Батурчик*
Компьютерная правка, оригинал-макет *А. А. Лысеня*

Подписано в печать 26.06.2013. Формат 60x84 1/16. Бумага офсетная. Гарнитура «Таймс».
Отпечатано на ризографе. Усл. печ. л. 4,07. Уч.-изд. л. 4,0. Тираж 100 экз. Заказ 300.

Издатель и полиграфическое исполнение: учреждение образования
«Белорусский государственный университет информатики и радиоэлектроники»
ЛИ №02330/0494371 от 16.03.2009. ЛП №02330/0494175 от 03.04.2009.
220013, Минск, П. Бровки, 6