6.5. За рамками интуитивно понятного интерфейса

Когда речь заходит о графическом интерфейсе (Graphic User Interface - GUI), наиболее популярное общее место – *интуитивно понятный*. Попробуем оценить его критически:

- предполагается, что GUI настолько очевиден и соответствует функционалу, что у пользователя не может возникать никаких специфических проблем, связанных с ним. Справедливости ради следует сказать, что если функционал системы является *интуитивно понятным*, а графический интерфейс ведет пользователя в соответствии с ним, то он также является *интуитивно-понятным*;
- в унифицированном процессе проектирования (UP) разработка GUI вообще является деятельностью второго плана в процессах функционального проектирования. Считается, что если функционал достаточно проработан в виде требований и сценариев, то проблематика GUI ограничена особенностями графического дизайна, эргономики и т.п.;
- требование *интуитивно понятный* уже само по себе является ересью, т.к. не может быть протестировано.

Получается, что и говорить не о чем, кроме как о *внешнем виде кнопочек* или модных трендах в виде плиток. Оставим в стороне внешний вид, как таковой, и попробуем оценить GUI с более широких позиций: насколько он помогает или мешает пользователю в достижении целей при его работе с системой. Здесь можно выделить ряд факторов, о стороны которых следует рассматривать GUI:

- производительность;
- человеческие ошибки;
- обучение;
- субъективное восприятие;
- запоминание, распределение пространства экрана, поиск, визуализация, навигация.

Данный материал представляет собой сжатое свободное изложение [6-1], подкрепленное собственными дополнениями, выводами и примерами.

Производительность

Производительность GUI или скорость работы с ним является комплексной оценкой всего процесса работы пользователя с системой через GUI и включает в себя следующие этапы:

- цель (обдумывание);
- последовательность действий (обдумывание);
- исполнение;
- восприятие;
- оценка результата.

Для оценки производительности GUI применяется методика, известная как **GOMS** (**G**oals, **O**perators, **M**ethods, and **S**election **R**ules – цели, операторы, методы и правила их выбора).

Способы исполнения действий также влияют на производительность (расположены в порядке ее увеличения):

- меню;
- горячие клавиши для опытных пользователей;
- пиктограммы, их основной недостаток - сложность подбора сочетания предмета на *картинке* с действием, назначенным пиктограмме;
- непосредственное манипулирование.

В психологии используется термин **фокус внимания.** Мы сосредоточимся на технологической стороне этого термина.

> **Фокус внимания –** концентрация внимания на некотором предмете, его поведении и управлении им. Восстановление фокуса внимания требует определенных временных и психологических затрат.

В GUI фокус внимания относится к объектам на экране, которые мы видим и можем ими манипулировать. Если в процессе работы мы переключаемся на другой объект, то для продолжения работы с первым необходимо не просто его появление на экране, но еще и восстановление контекста исполняемого действия при восстановлении фокуса внимания, что включает в себя:

- исполняемое действие;
- шаг, на котором остановился пользователь;
- введенные параметры;
- текущий фокус ввода - позиция курсора.

Замечание по теме. Здесь напрашивается очевидная аналогия с состоянием процессора при многозадачной или многопоточной работе.

Длительность физических действий. Любое действие может быть либо быстрым, либо точным. Время достижения цели обратно пропорционально размеру цели и дистанции до цели. Сказанное справедливо при манипулировании любыми объектами в GUI. Соответственно, чем короче перемещения при манипулировании, тем выше производительность. ***Рекомендации по теме:***

- контекстное меню (расстояние минимально);
- диалоговое окно по месту элемента управления;
- граница экрана как псевдо-кнопка. При *залипании* курсора на границе экрана появляется большой элемент GUI, который дает быстрое позиционирование и не требующее точности.

Длительность реакции системы. Если выполняемые системой операций приводят к ощутимым задержкам, то необходима корректная оценка времени ее исполнения, в течение которого не требуется вмешательства пользователя. ***Рекомендации по теме:***

- перед началом длительного процесса все данные должны быть получены сразу, недопустим запрос дополнительных данных после начала выполнения операции;
- установка тайм-аут на всплывающие окна с подтверждением операции, по истечении интервала времени по умолчанию принимается положительный ответ;
- реализация реального прогресс-индикатора. Типичной ошибкой является игнорирование финальных операций, когда индикатор со значением «осталось 0 секунд» висит достаточно долго.

Адаптация. Желательно, чтобы система подстраивалась под контекст, в котором работает пользователь, что позволяет минимизировать число выполняемых в GUI действий. ***Рекомендации по теме:***

- разделение настроек по умолчанию для повторяющихся и эпизодических действий. Например, текущий каталог сохранения файлов меняется при сохранении в новый каталог более двух раз подряд, однократные эпизодические сохранения не изменяют принятого умолчания;
- разделение настроек по умолчанию для разных типов операций. Например, сохранение отчетов и экспорт данных производится в каждом случае свой текущий каталог;
- адаптация окон настройки операции в зависимости от *истории* работы, последовательная развертка окон настроек с различной степенью подробности. Например, первое окно - кнопка ОК и ссылка «параметры», второе окно - кнопка ОК, последние измененные или часто используемые параметры, третье окно - кнопка ОК и все параметры.

Человеческие ошибки

Человеку свойственно ошибаться. Дружественный характер GUI предполагает, что система не является ментором, уязвляющим пользователя, но предлагает варианты исправления. Ошибки могут быть вызваны различными причинами:

- пробелы в знаниях предметной области;
- оЧеПятки;
- моторные ошибки, связанные с неточным манипулированием мышью;
- снижение внимания, пропуск/игнорирование предупреждений.

Замечание по теме. Эта тема перекликается с программными ошибками (п.8.2) в том смысле, что системе тоже свойственно ошибаться. И в том и в другом случае ошибки желательно предупреждать, сообщать пользователю не столько о самой ошибке, сколько о способе ее исправления.

Рекомендации по теме. Меры предотвращения ошибок:

- обучение пользователей в процессе работы;
- снижение требований к внимательности;
- повышение разборчивости и заметности индикаторов;
- снижение чувствительности системы к ошибкам:
  - блокировка потенциально опасных действий пользователя, использование **пред- и пост-подтверждений**;
  - проверка системой всех действий пользователя перед их принятием, ограничение диапазона и формата данных - **выбор вместо ввода**;
  - адаптация системы к действиям пользователя - **умолчания, запоминание статистики и истории работы**.

Способы предотвращения опечаток:

- использование выбора вместо ввода, вывод границ допустимого диапазона, отображение недопустимого ввода сменой цвета / курсивом;
- учет фактора потери бдительности. Например, не стоит делать кнопку ОК кнопкой по умолчанию;
- использование команды явного **разблокирования** вместо **подтверждения операции.**

Качественный дизайн GUI предполагает исправление во время совершения действия, либо исключает возможность их появления

Обучение и самообучение

Под обучением понимается не только система подсказок и программной документации. В более широком смысле – это учет всевозможных факторов, позволяющих использовать возможности системы пользователями с различными уровнями квалификации и знаниями предметной области. Для пользователя любая система ценна возможностью решения с ее помощью текущих задач, желательно с минимальными затратами на вхождение. При этом само по себе наличие определенных возможностей еще ни о чем не говорит, воспользоваться ими могут помешать:

- незнание о существующих возможностях – вопрос «что можно сделать?»;
- сложность поиска элементов управления – вопрос «где это искать?»;
- неумение ими пользоваться – вопрос «как это сделать?».

Замечание по теме. Разнообразие уровня знаний о системе и предметной области, с которой она работает, порождает разнообразие вариантов и способов обучения. Например, для обычного редактора изображений можно выделить такие категории пользователей:

- имеет поверхностное представление о предметной области, использует для выполнения разовой работы, например, поменять цвета на картинке или сжать ее;
- имеет поверхностное представление о предметной области, использует для решения ограниченного ряда простых задач;
- продвинутый пользователь - профессионал, знающий систему, желающий получить с ее помощью качественный продукт, использующий систему как рабочий инструмент.

Если для первого нужен перечень описания решений типовых задач по принципу, *какие кнопки давить*, то для второго нужно либо упрощенное описание, либо система подсказок в интуитивно-понятном интерфейсе. Для третьего нужны подборки материалов по художественному дизайну, адаптированные к системе.

Для создания целостного образного представления о системе используются следующие средства:

- **ментальная модель** – логическое или образное описание организации системы, с которой работает пользователь;
- **метафора** – представление интерфейса в виде аналога, знакомого пользователю;
- **аффорданс** – очевидность функционала, исходящая из визуального образа объекта;
- **стандарт** – общепринятые правила обозначения действий и свойств.

Метафора - интерфейс системы воспроизводит **предметную область** (полиграфия, звукозапись, типографическая верстка) в этом случае графический интерфейс моделирует общеизвестный реальный предмет (рис.6-23).

Виды метафоры могут быть различными. Она может охватывать как систему в целом, так и отдельную ее часть, например, красная лампочка как индикатор ошибки или опасности. Предмет метафоры может быть придуман разработчиком и не иметь никакого отношения к предметной области.

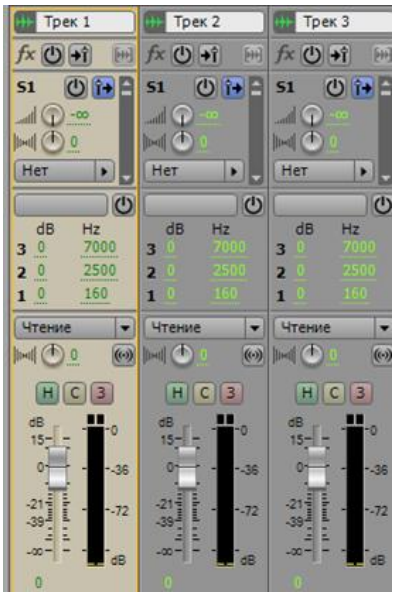

Рис.6-23. Метафора звукового редактора – режиссерский пульт

Несмотря на свою привлекательность, использование метафор таит в себе проблемы следующего свойства:

- сложность подбора;
- метафора может ограничивать систему;
- со временем метафора может стать архаичной, если сам предмет метафоры перестает использоваться, например, печатная машинка;
- метафора должна быть общеизвестной;
- метафора должна покрывать весь функционал, новый функционал сложно вписать в имеющуюся метафору.

Аффорданс. Термин из области психологии. Обозначает свойство предмета, позволяющее сразу понять, как этим предметом пользоваться. Как правило, аффорданс ограничен визуальной компонентой, звуковые, тактильные, и пространственные ощущения, для которых принцип аффорданса может быть применен, в графическом интерфейсе не используются. С явной натяжкой к аффордансу можно отнести имена файлов типа *ReadMeFirst.txt* или *RunMe.exe*.

Несмотря на свою очевидность, аффорданс не всегда бывает однозначен именно в силу возможного различного представления об этой очевидности. Иногда логика представлений может быть весьма парадоксальна. Например, рассмотрим соответствие позиций тройного выключателя и расположения санузла, ванной и кухни в квартире (рис.6-24). Расположение дверей относительно выключателя: санузел и ванна рядом с выключателем слева и справа, кухня - на некотором расстоянии справа. Можно предполагать такие варианты аффорданса:

- соответствие в порядке следования слева направо;
- поскольку санузел и ванная расположены рядом и симметрично, то логично расположить позиции выключателей по краям. Выключатель для кухни посередине, т.к. она является *третьим лишним*;

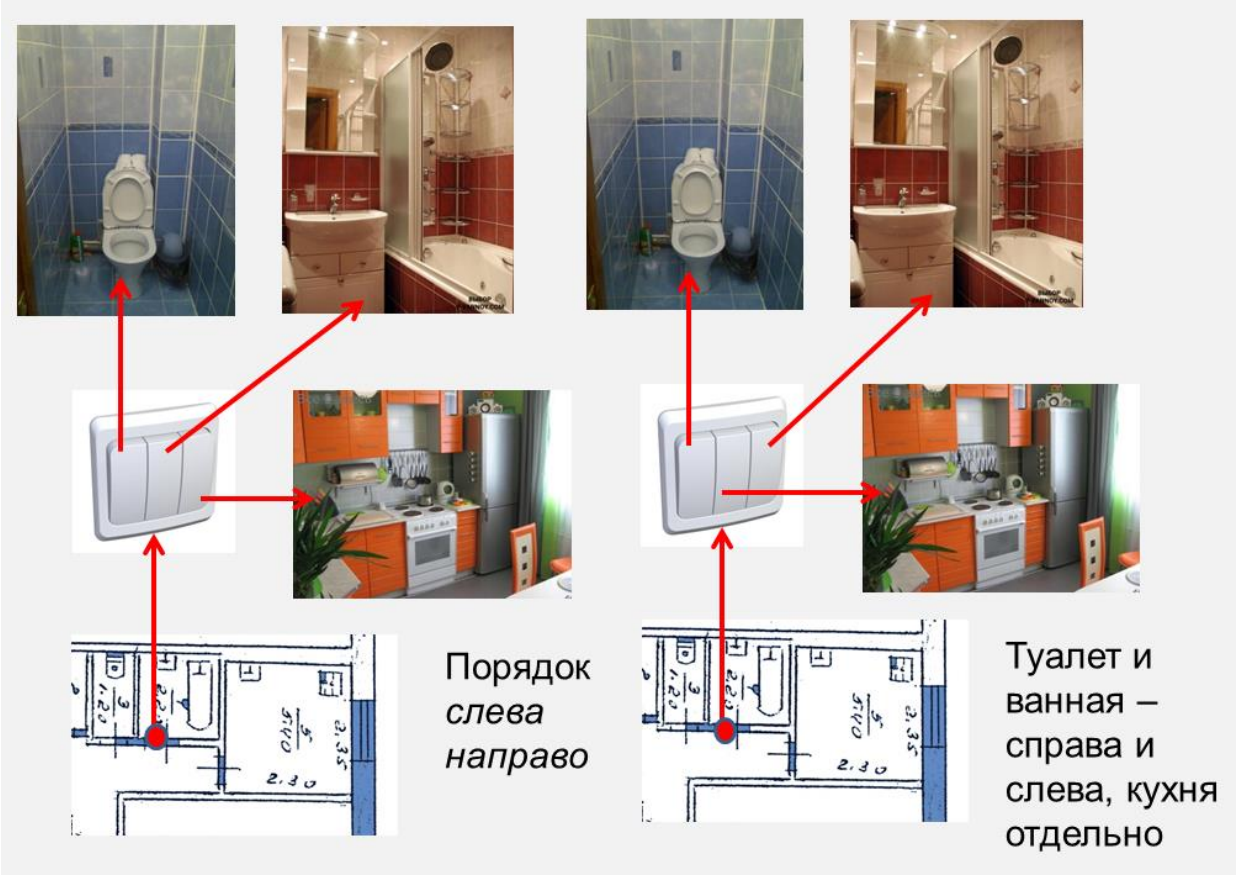

Рис.6-24. Неоднозначность логики аффорданса

Замечание по теме. Аффорданс основан на логике или точке зрения, принимаемой по умолчанию, очевидной. Но точка зрения может быть парадоксальной, иногда обратной общепринятой. Приведем примеры ошибочных представлений из области терминологии:

- если во главу угла ставить не программу, а файл, то запись данных в файл – это *ввод*, в чтение – *вывод* данных из файла;
- обычно при работе с сервером термин префикс *down* подразумевает получение данных с сервера, а *up* – передачу данных на сервер, т.е. образно он подразумевается *наверху*. Так обычно и изображается на схемах и рисунках. В то же время в программах обмена данными с мобильными телефонами заливка данных на сервер иногда фигурировала под термином *down*, поскольку телефон мыслился, видимо, как вершина мироздания.

Стандарт. Сюда относятся не только стандарты в общепринятом смысле, но также и общепринятые элементы графической, цветовой и прочей информативности, т.е. по существу, тот же аффорданс. Например:

- стандарт графического дизайна приложений;
- цветовая гамма: красный/желтый/зеленый – недоступен/подключение/ доступен, оранжево-черный – обозначение опасности;
- общепринятые пиктограммы.

До сих пор речь шла об априорных знаниях и представлениях о системе. Собственно информационное наполнение справочной системы включает в себя следующие варианты местоположения справочных данных и способов их получения:

1. бумажная документация, электронные документы, видео-уроки;
2. электронный структурированный документ – система помощи (help);
3. отдельные экранные формы, диалоги;

4. контекстное меню, всплывающие подсказки;
5. управляющие элементы экрана, строка состояния.

В системе документации приняты следующие виды обучающих материалов:

- **базовая справка** – назначение системы, используется при первом входе (1,3);
- **обзорная справка –** обзор функций системы (1,3);
- **справка предметной области –** сведения о предметной области и профессиональных приемах работы (1);
- **процедурная справка –** способы реализации основных функций системы (2, желательна привязка к графическому интерфейсу);
- **контекстная справка –** назначение элементов управления (4);
- **справка состояния –** (4,5);
- **сообщения об ошибках** - (3,4,5).

Справочные материалы желательно максимально вписывать в контекст работы

Запоминание, распределение пространства экрана, поиск, визуализация, навигация

Через графический интерфейс проходит вся информация, которую пользователь получает от системы. И здесь очень важно, как она будет структурирована для представления и в какой форме визуализирована. Формально эти вопросы не относятся к функционалу, но они крайне важны.

Запоминание. Реально человек способен эффективно манипулировать только предметами, непосредственно находящимися в поле зрения. Объем кратковременной памяти ограничен 7±2 единицами неассоциированных данных, т.е. данных, при запоминании которых не возникало образных ассоциаций. При проектировании графического интерфейса необходимо свести к минимуму необходимость такого запоминания. В структуре графического интерфейса объекты разделяются по уровню их доступности и необходимости пользователю задействовать кратковременную память:

- непосредственно видимые;
- прямо доступные через видимые ассоциируемые элементы - закладки, иконки;
- выбираемые через видимые ассоциируемые элементы - выпадающие списки, всплывающие окна;
- находящиеся в известной пользователю цепочке обращений, например, открытие файла в диалоговом окне через меню File;
- находящиеся в неизвестной пользователю цепочке обращений, например, неизвестный пользователю способ настройки или параметр.

Последний пункт имеет отношение к фактору **обучения**. Остальные – к фактору **производительности** и к определенному в нем фокусу внимания.

Распределение пространства экрана. Для многих приложений графическое пространство экрана является критическим ресурсом, который надо умело распределять между отображаемыми данными. Одной из трудно решаемых проблем является *отсеивание лишней информации***.** Под лишней может пониматься обнаруженная при поиске ненужная, устаревшая или более не используемая информация. Обычно в процессе работы с программой количество активных элементов - открытых окон, закладок и т.п. постоянно увеличивается, а закрывать или удалять их приходится вручную. Средств сбора такого *интерактивного мусора* обычно не предусмотрено.

Интерактивный мусор – созданные в процессе работы, но не используемые более элементы GUI (окна, закладки, иконки)

Так бывает. В системе разработки открытые файлы отображаются в виде закладок с такими правилами (рис.6-25):

- количество видимых закладок в пределах десятка, размер закладки зависит от длины имени;
- закладки отображаются в порядке открытия файлов, повторное обращение не меняет их порядка в списке;
- для получения скрытой закладки необходимо *промотать* список в панели закладок или открыть полный список отдельным кликом.

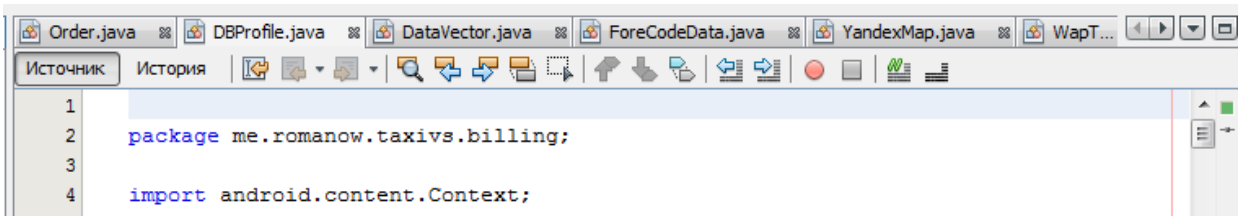

Рис.6-25. Список открытых файлов проекта в IDE NetBeans

При постоянной работе с большим проектом приходится периодически закрывать ненужные окна. Поэтому логично было бы иметь средства, отслеживающие и закрывающие долго не используемые или однократно используемые окна,

Ограничивайте и своевременно утилизируйте интерактивный мусор

Визуализация, поиск, навигация. Визуализация – представление данных пользователю в форме, удобной для восприятия, оценки и манипулирования (рис.6-26). Пользователю нужны не данные, как таковые, а их анализ для достижения некоторой цели, например, для освобождения места на диске путем удаления очень больших файлов или очистки каталогов от устаревших данных.

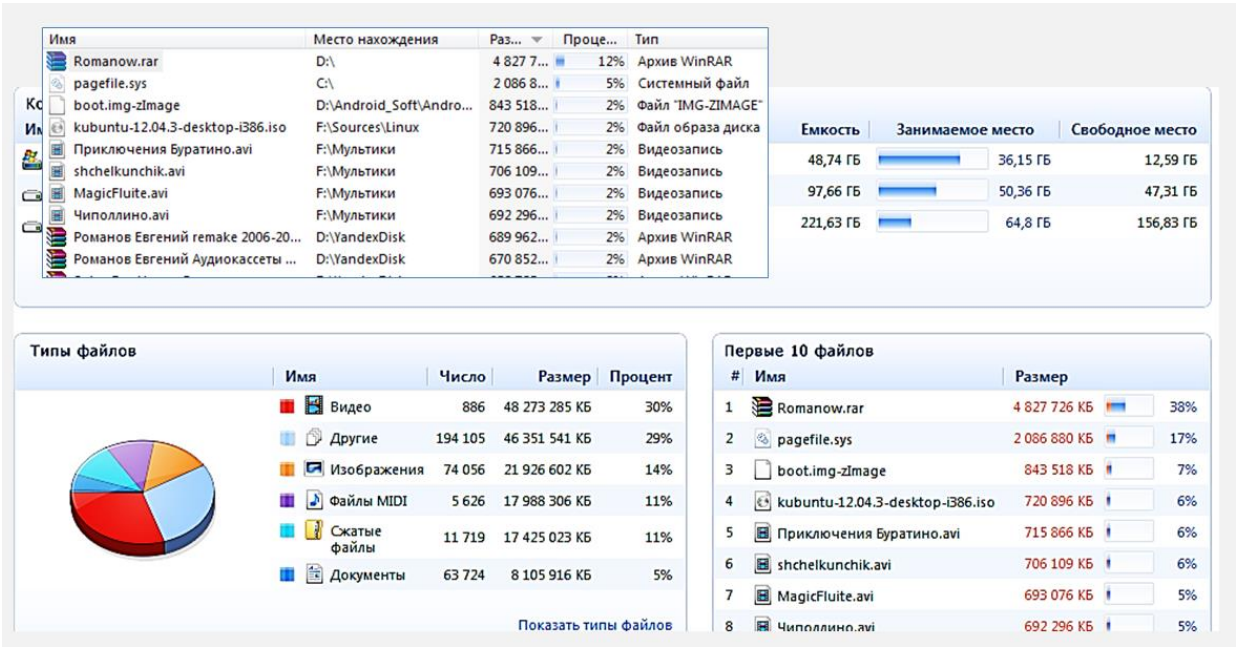

Рис.6-26 Визуализация в TuneUp Utilities

Стоп-кадр: популярный кадр 60-70 годов: ученые рассматривают распечатку результатов в виде «простыни» с колонками цифр.

Многообразие способов визуализации лучше всего проявляется на числовых данных. Зачастую важным является на само значение, а его качественная оценка по каким-то критериям или шкалам. Тогда лучше использовать формат представления, удобный для оценки. Рассмотрим в качестве примера вывод данных о размерах файлов. Возможные варианты:

- относительное значение, в процентах к максимальному значению, общему объему ресурса, к общей сумме значений;
- порядок значения, логарифмическая шкала с ограничением количества значащих цифр (размер файла в виде 376, 12Кб, 186Мб);
- превышение значения относительно порога;
- качественная шкала, размер файла изображения - крохотный, маленький, средний, большой, огромный;
- группирование малых значений, например: остальные 1478 файлов, 36Мб.

Способ визуализации значения также может быть отличен от обычного числового:

- линейный индикатор - полоска;
- пиктограмма - количество, цвет, размер, пропорциональный значению;
- точка в системе координат - двумерные данные;
- диаграмма;
- распределение (гистограмма) – группировка для больших наборов данных;
- индикатор цветности - яркость и насыщенность, пропорциональные значению.

Замечание по теме: научная дисциплина **разведочный анализ данных** - использование средств визуализации в сочетании со средствами статистической их обработки с целью выявления закономерностей, аномалий, группировки и подобного анализа. Активно задействует интуицию исследователя.

Поиск и навигация. По аналогии с визуализацией исходить следует из целей, которые преследует пользователь.

Так бывает. Типичный лог для фиксации ошибок/событий от мобильных приложений (рис.6-27):

- поле с номерами страниц лога находится внизу, так что для перехода к следующей странице приходится прокручивать страницу до конца;
- обычно поиск производится по конкретной дате или диапазону дат. По номерам страниц сложно предполагать о датах содержищихся в них записей. Кроме того, что не все номера страниц лога выводятся в списке.

Рис.6-27. Две ошибки поиска-навигации в логе ошибок

Так бывает. При отображении больших таблиц обычно используется прокрутка всего содержимого таблицы. Это создает существенные неудобства, ибо заголовки таблиц должны быть видны всегда, поскольку с ними ассоциируется остальное содержимое. В desktop-приложениях логичнее использовать способ отображения таблиц, который планирует двумерный скроллинг только для областей данных, а области заголовков прокручивает синхронно в одном измерении. В мобильных приложениях можно просто отмечать выбранные строку и столбец (рис. 6-28).

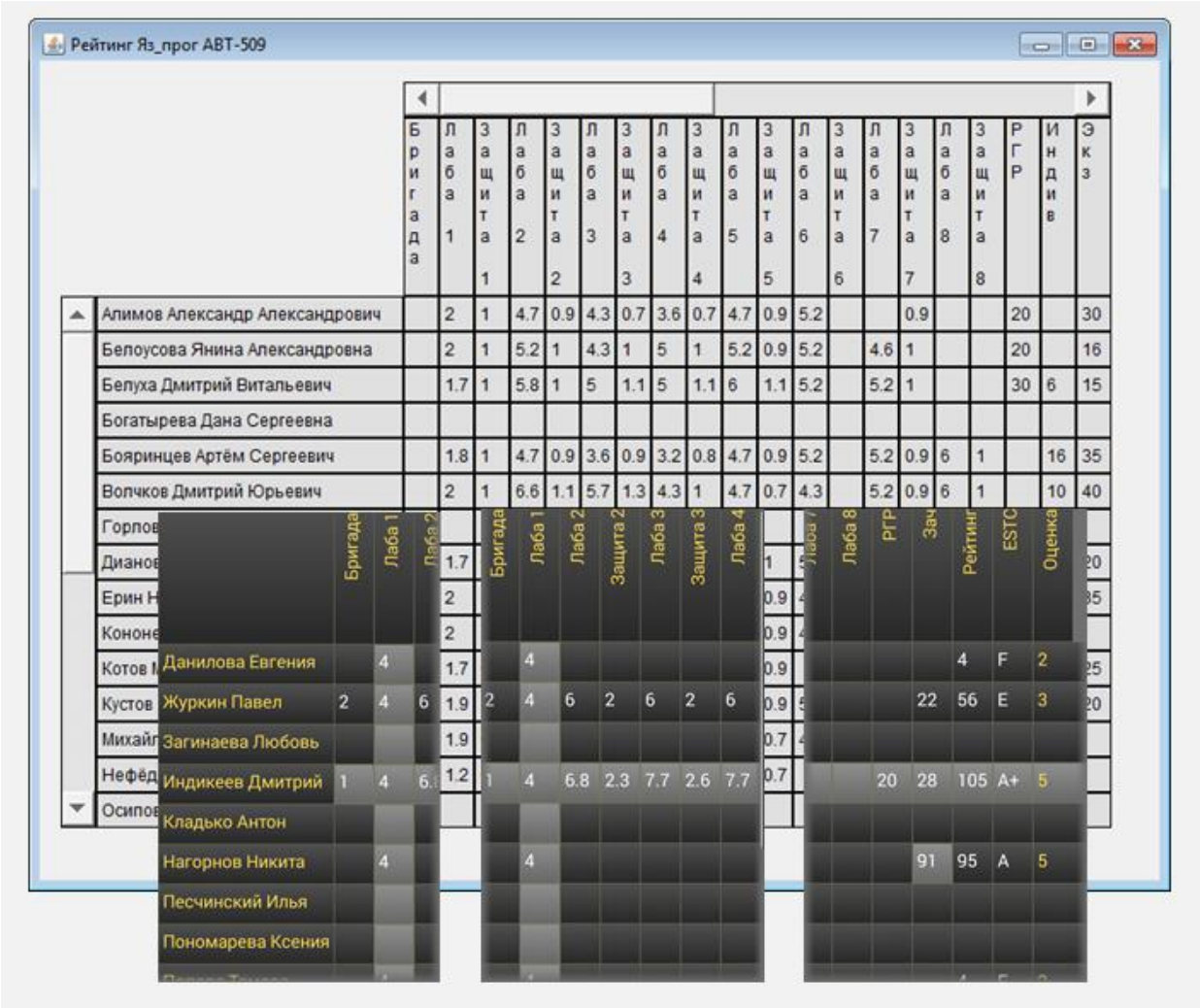

Рис.6-28. Прокрутка таблиц с сохранением заголовков

Субъективное восприятие

И наконец, мы подходим к собственно внешнему виду интерфейса, графическому дизайну и особенностям его субъективного восприятия пользователем. Этот фактор включает в себя:

- социально-психологическое восприятие - мода, «крутизна»;
- технический дизайн - сочетание эстетики с технологичностью;
- психологическое ощущение комфорта при работе.

Социально-психологические аспекты. Графический дизайн приложений сильно подвержен моде, может быть ориентирован на определенные социальные группы, которые составляют значительную часть пользователей, зачастую должен отражать требования брендов, которые он продвигает и т.п.. Все это имеет весьма отдаленное отношение к программной инженерии, поэтому позволю дать только одну дилетантскую

рекомендацию: чувство меры и знание истории технологического дизайна. Иначе возможны ситуации, когда внешний вид графического интерфейса будет вызывать у некоторых пользователей непредвиденные ассоциации. Несколько примеров:

- ассоциация по технологическому дизайну (рис.6-29). Модный ныне минимализм графического дизайна перекликается с технологическим минимализмом 60-х годов - аналог «хрущевок» в технологии предметов домашнего обихода;

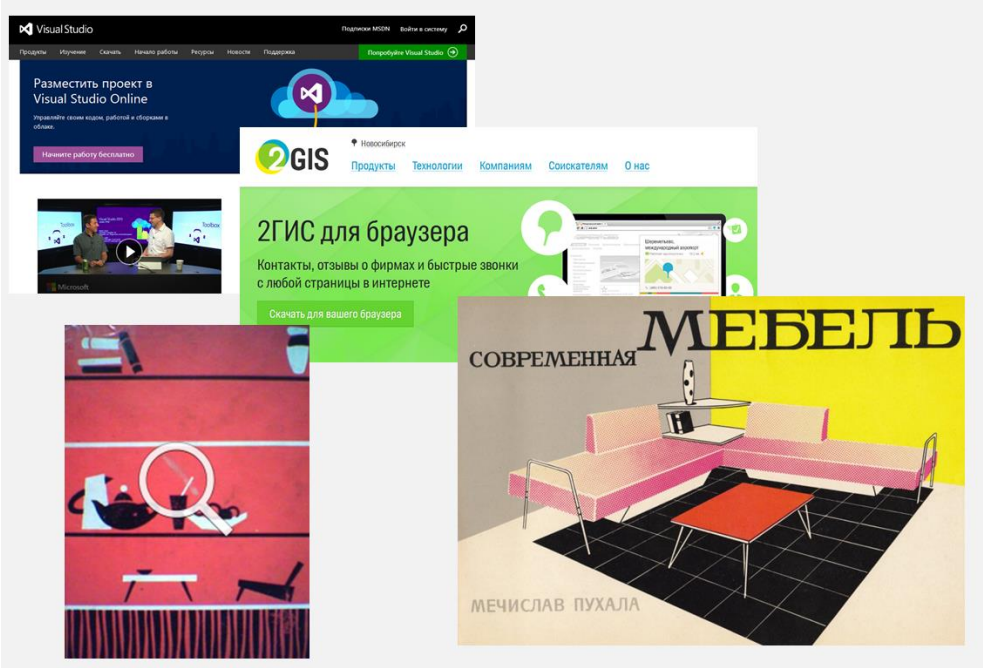

Рис. 6-29 Ассоциация с технологическим дизайном 60-х годов

- ассоциация по цветовой палитре (рис.6-30). Графический дизайн в мягких пастельных тонах перекликается с фотографиями 50-х годов прошлого века. Особенности последних обусловлены слабой цветовой чувствительностью тогдашней фотопленки;

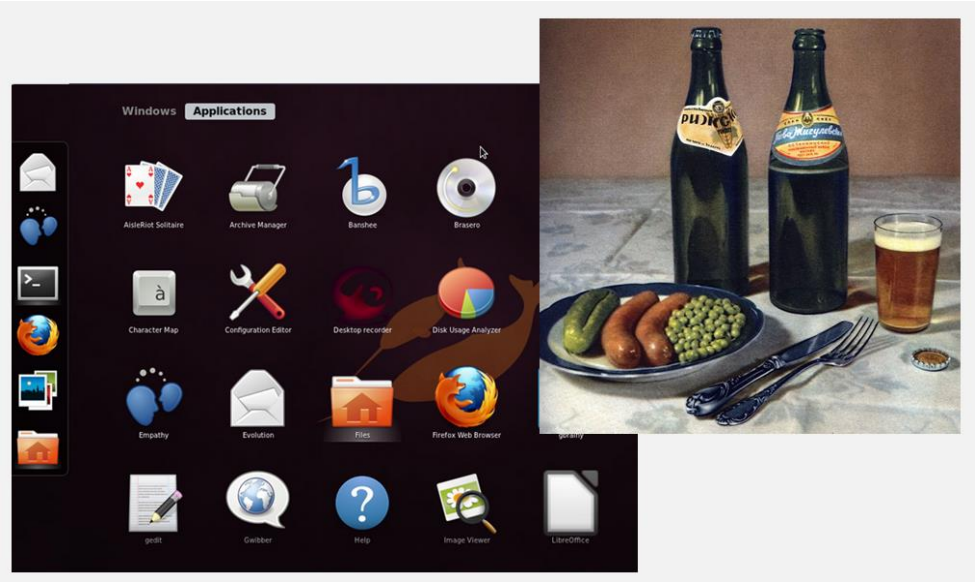

Рис. 6-30 Ассоциация по цветовой палитре

- чувство меры в цветовой гамме (рис.6-31). Многие бренды используют определенные тона или их сочетания, излишне яркие краски отвлекают от содержания.

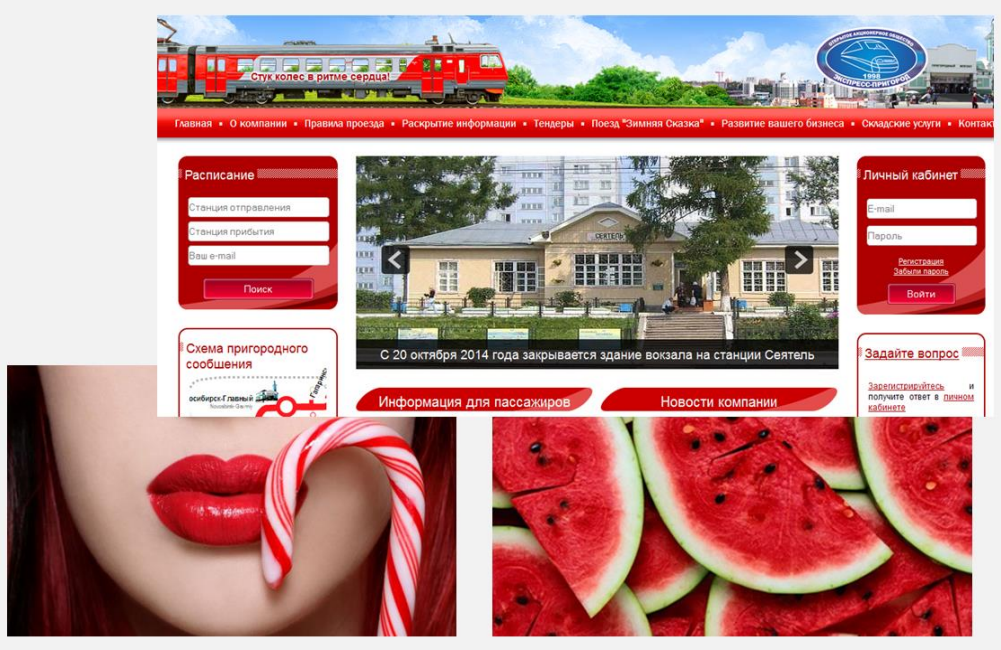

Рис. 6-31 Чувство меры в цветовой гамме

Принципы технического дизайна интерфейсов. Если социально-психологический фактор использует интерфейс как дополнительный фактор привлечения внимания, то технический дизайн требует от него в точности обратного – он должен быть максимально незаметен, прозрачен и подчинен функционалу (рис.6-32), а именно:

- интерфейс – не самоцель, он незаметен;
- интерфейс функционален и информативен;
- интерфейс – предмет длительного использования;
- интерфейс технологичен, он обеспечивает производительную работу, минимизирует ошибки;
- интерфейс гармоничен - все элементы соизмеримы, соразмерны, выполнены в едином стиле.

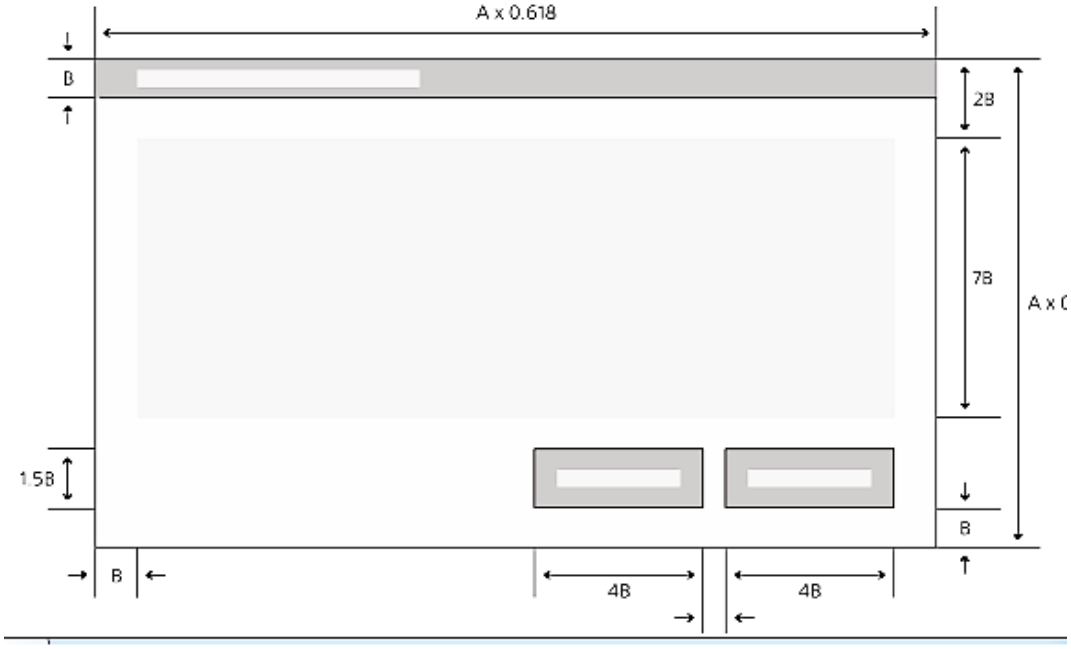

Рис. 6-32. Технический дизайн GUI: прозрачность, гармоничность

Отсюда самые простые рекомендации:

- исключение ярких цветов и острых углов;
- контраст за счет пустых пространств между элементами. По аналогии с музыкой – контраст обеспечивается не за счет громкости, а за счет паузы и звучания;
- легкость и прозрачность интерфейса;
- исключение крикливости и навязчивости;
- визуальные закономерности: масштабная сетка, *золотое сечение* в пропорциях 0.618 x 0.382, пропорциональность элементов.

Субъективное ощущение комфорта. Не всегда качество графического интерфейса можно выразить измеряемыми параметрами. Субъективное ощущение дружественности интерфейса складывается, в том числе, из ряда психологических факторов:

- субъективное ощущение скорости работы: заполнение пауз фоновыми действиями, разбиение действий на более мелкие;
- чувство контроля над системой:
  - ограничение и контроль за потенциально опасными действиями;
  - *красная/зеленая лампочка* – визуализация состояния соединений, доступности ресурсов;
  - корректная система контроля ошибок: предупреждение ошибок, указание границ действий, выбор вместо ввода, исправление в процессе ввода, сообщения о причине ошибки и способах исправления;
- самовыражение - возможность персональной настройки программы;
- разумная система идентификации и защиты: выбор логина вместо ввода - coockies, выпадающий список, ввод пароля открытым текстом, запоминание пароля на ограниченный срок. Обычно системы защиты вносят значительный дискомфорт, т.к.действия по предотвращению потенциального вреда не воспринимаются как необходимые, сложные пароли необходимо запоминать или хранить отдельно и т.п..

Влияние GUI на процесс разработки. Проектирование от GUI

Как уже было сказано в начале, в унифицированном процессе проектирования разработка графическогно интерфейса вообще является деятельностью второго плана в процессах функционального проектирования. Первичными являются варианты использования, сценарии и требования как способы описания функционала.

Соответствие между функционалом и графическим интерфейсом не является простым отображением элементов первого на компоненты второго. Например, сценарий может быть реализован как последовательность вызывающих друг друга диалогов, либо как действия над списками и кнопками в одном окне в более-менее свободном порядке. Требование визуализации состояния может быть реализовано в виде всплывающего сообщения об изменении состояния, либо в виде постоянного индикатора в основном окне.

Рассмотрим несколько типовых вариантов организации графического интерфейса с точки зрения их связи с функционалом. В первом варианте (рис.6-33) мы имеем дело с набором инструментов (действий), которые можно применять к объекту предметной области - в данном случае, это изображение.

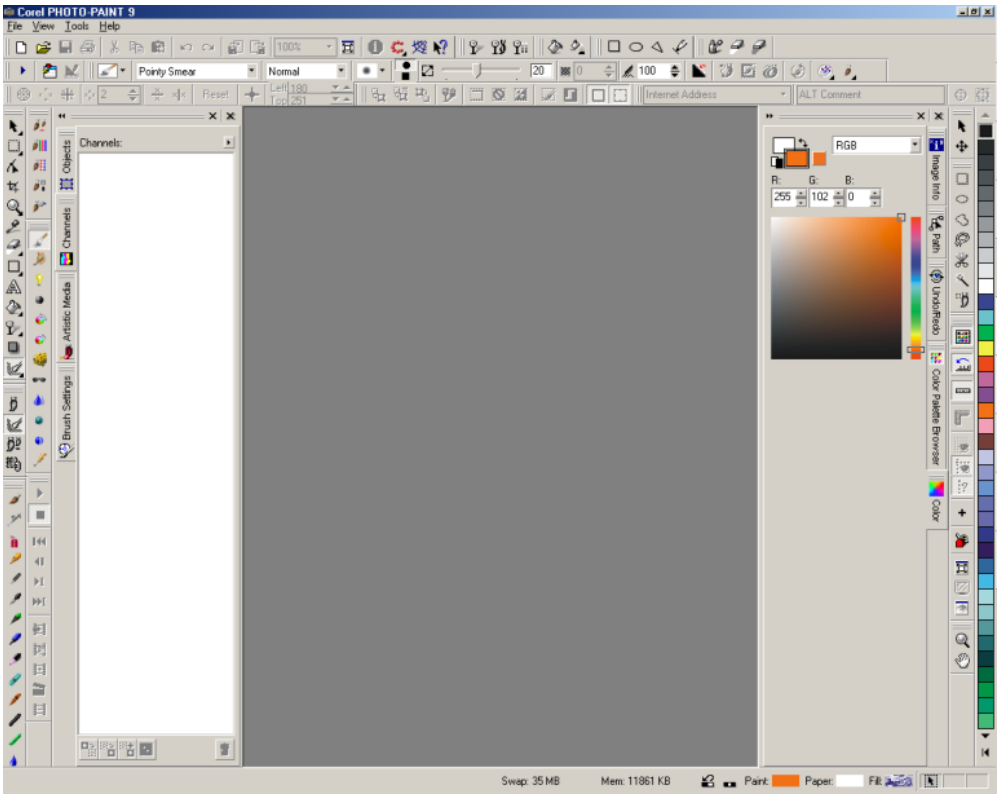

Рис.6-33. Интерфейс - *набор инструментов*

В более сложном случае предметная область представлена в виде более сложной модели (рис.6-34).. Например, звуковой редактор работает с треками, в каждый из которых может быть загружен одни или несколько звуковых файлов. Для каждого из них устанавливается цепочка эффектов обработки звука и набор параметров - громкость, панорама. Результат микшируется в отдельный трек с соответствующей постобработкой

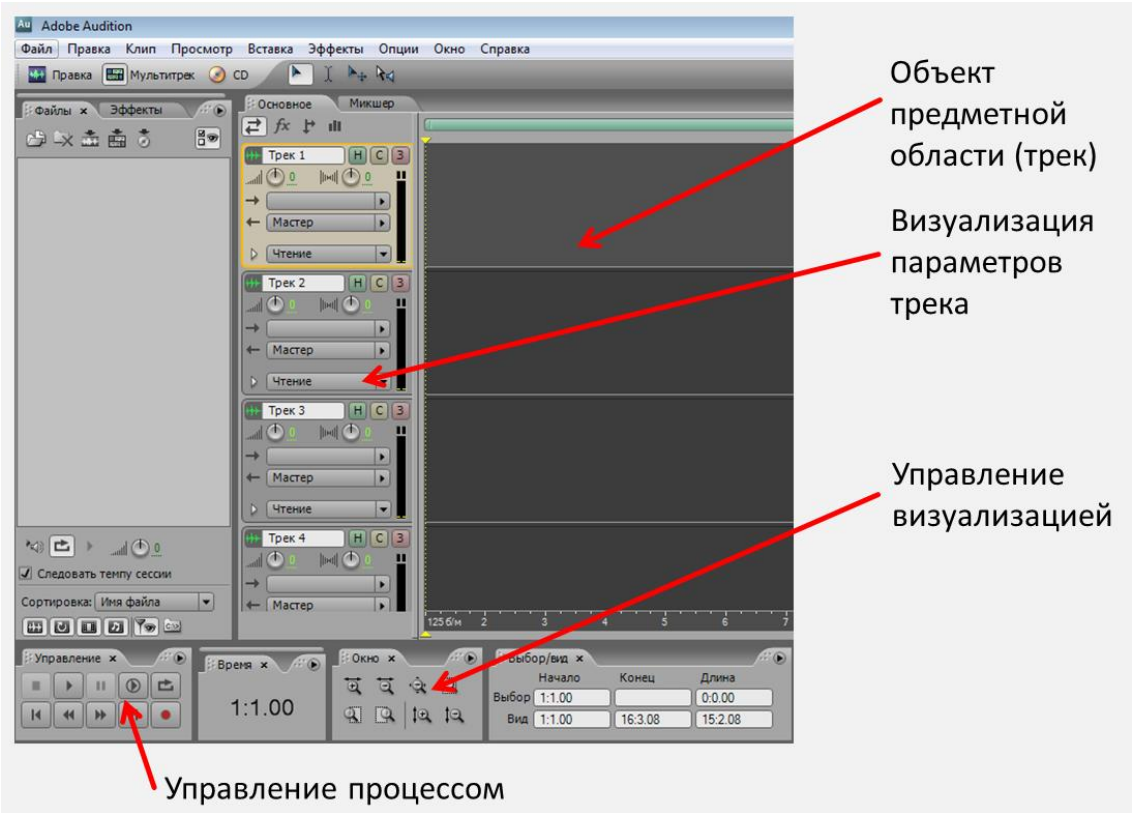

Рис.6-34. Интерфейс - *система окон над предметной областью*

Графический интерфейс представляет собой систему окон, в которые проецируется текущее состояние отдельных компонент предметной области или производится управление ими.

В приложениях, не рассчитанных на широкую публику, и в программных прототипах, где нет смысла в отдельном проектировании графического интерфейса, уместен вариант типа *кабина самолета* (рис. 6-35), где все элементы управления находятся в одном окне и одинаково доступны, а сценарии пользователь держит в голове.

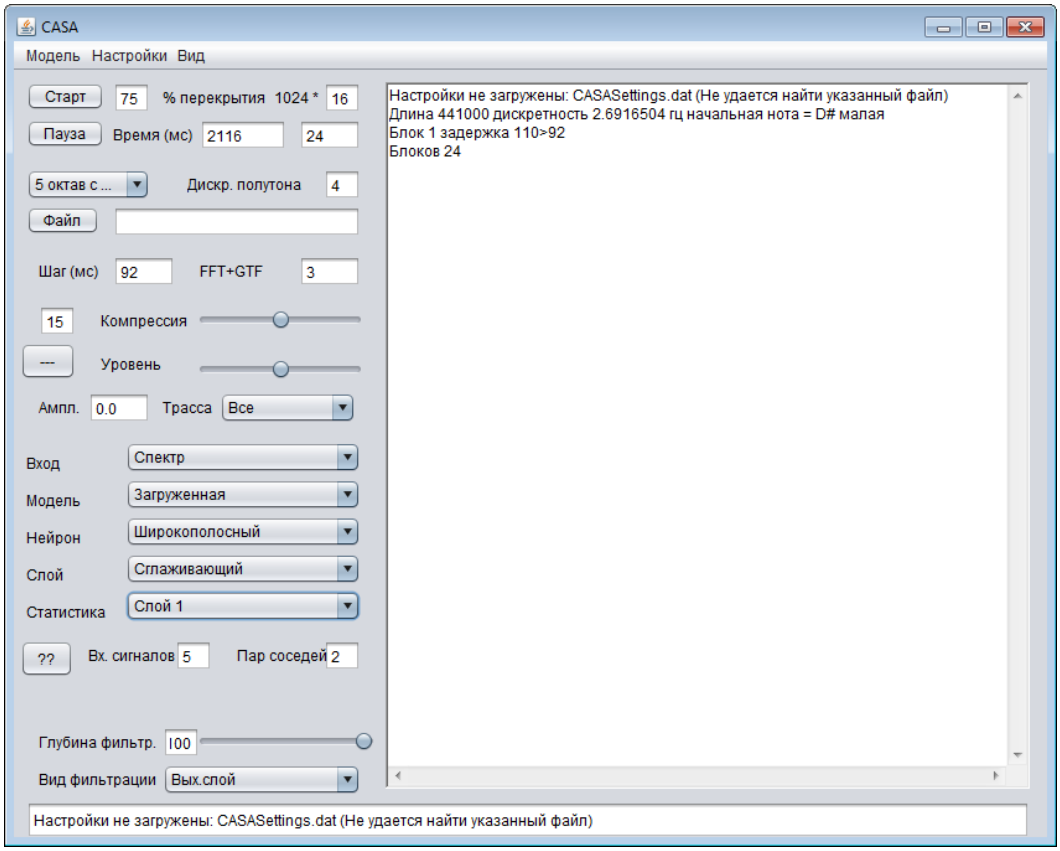

Рис.6-35. Интерфейс - *кабина самолета*

В мобильных приложениях ввиду ограниченности размера экрана сценарии, предписанные функционалом, поддерживаются цепочкой окон, в каждом из которых выполняется одно или несколько действий и находится ограниченное количество элементов управления (рис. 6-36).

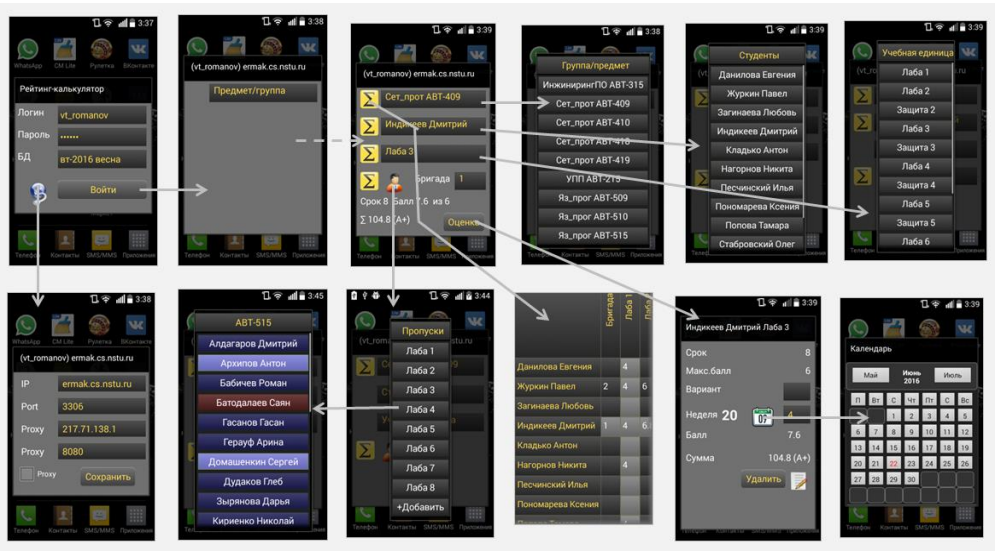

Рис.6-36. Интерфейс - *сценарии, ведомые интерфейсом*

Ввиду таких достаточно запутанных соотношений между графическим интерфейсом и функционалом, графический интерфейс, как таковой, не может полностью заменить функциональное описание. Тем не менее, есть несколько принципиальных мнений разработчиков по этому поводу:

- прототип графического интерфейса в виде набора экранов и есть описание функционала (заблуждение);
- роль графического интерфейса настолько важна для успеха проекта, что его нельзя прицеплять к готовому функционалу;
- характер взаимоотношений между предметной областью, функционалом и графическим интерфейсом позволяет рассматривать последний как одну из спецификаций функционала. Приведенные выше интерфейсы *сценарий, ведомый интерфейсом* и *система окон над предметной областью* являются иллюстрацией этого положения;
- опытные разработчики графического интерфейса фактически выполняют функциональное проектирование системы. Особенно такое мнение распространено у разработчиков web-приложений и сайтов.

Вариант разработки функционала системы *от графического интерфейса* [6-1] не отвергает основные элементы бизнес- и системной аналитики. Первоначально на каждый функциональный модуль системы (единицы поведения) создается отдельная модель. В качестве инструментов описания процессов и проектирования могут использоваться различные модели (рис. 6-37): диаграммы потоков данных (DFD), UML-диаграммы деятельности, UML-диаграммы устойчивости – для описания бизнес процессов. Собственно на этапе проектирования применяются неканонические диаграммы объектов графического интерфейса - окон, диалогов и переходов между ними или канонические UML-диаграммы оконных классов.

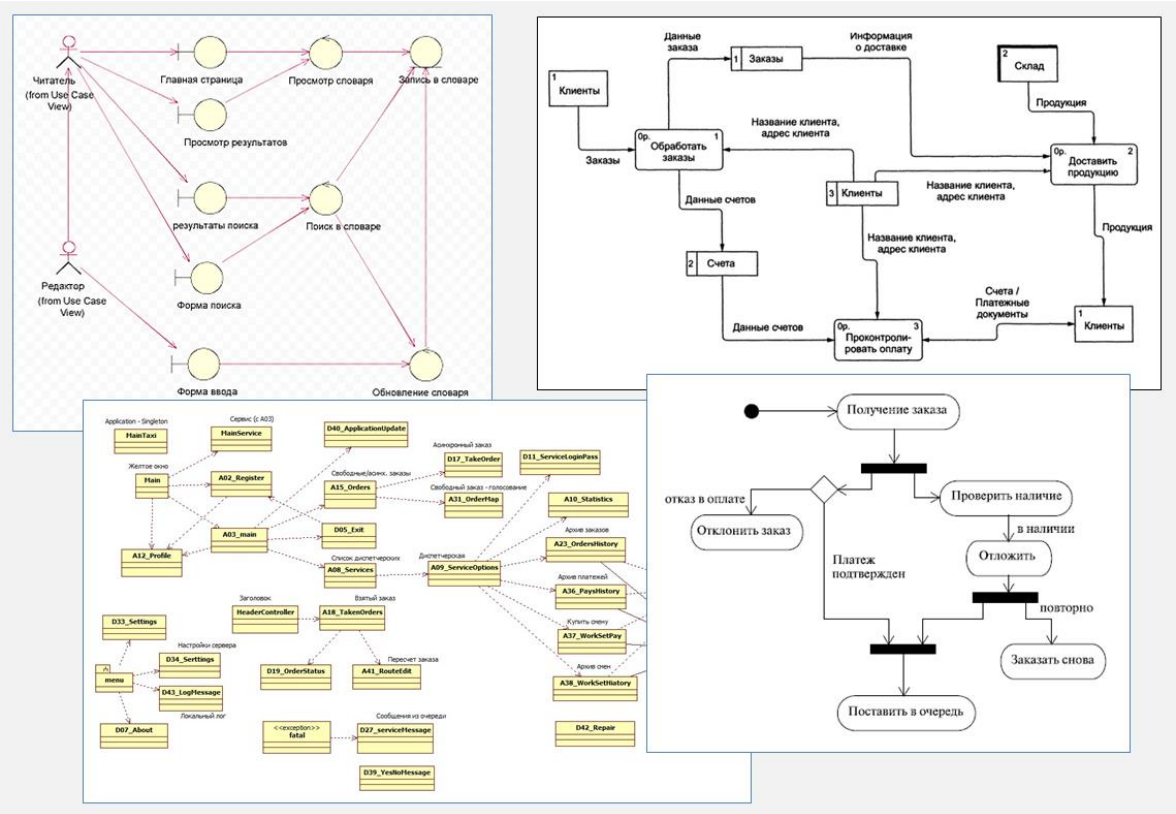

Рис.6-37. Модели, используемые при проектировании *от графического интерфейса*

Проектирование включает в себя следующие этапы:

- анализ функциональности системы, словесное или формальное описание бизнес-процессов и системной аналитики без привязки к графическому интерфейсу;
- разработка структурной схемы графического интерфейса с привязкой элементов бизнес-процессов к ее компонентам - экранам, всплывающим сообщениям, диалоговым окнам, строкам состояния (рис.6-38);
- верификация и модификация структурной схемы на основе верификации и анализа бизнес-процессов;
- детальное проектирование отдельных прототипов графического интерфейса, например, с использованием метрики скорости работы (GOMS);
- разработка глоссария и согласование прототипов;
- сведение всех прототипов в общую модель (рис.6-39).

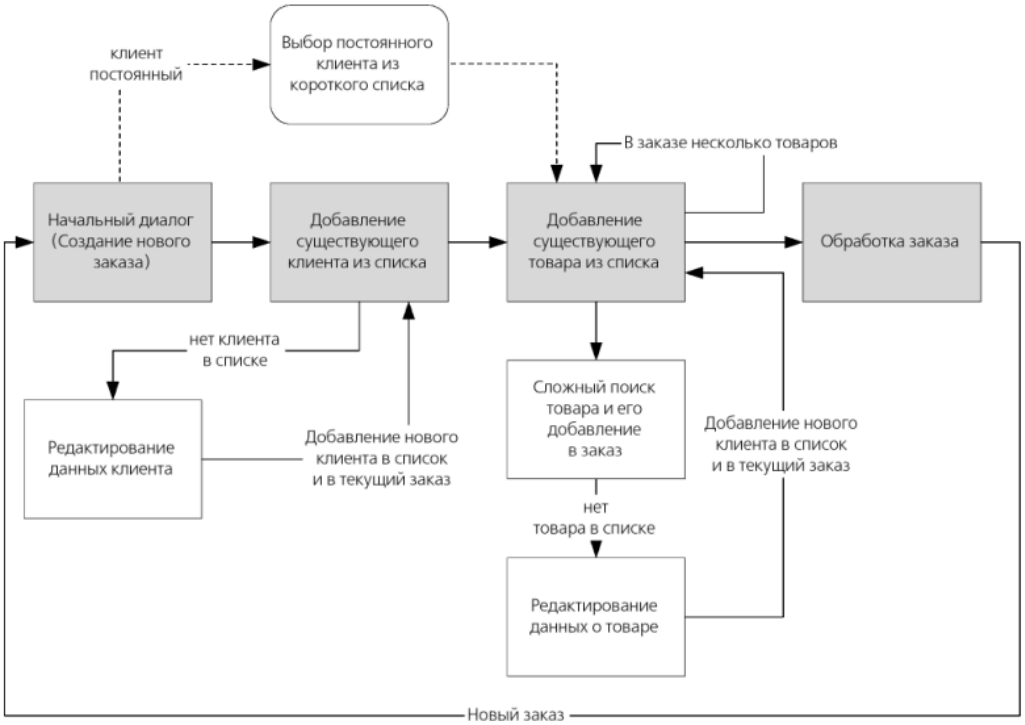

Рис.6-38. Структурная схема графического интерфейса

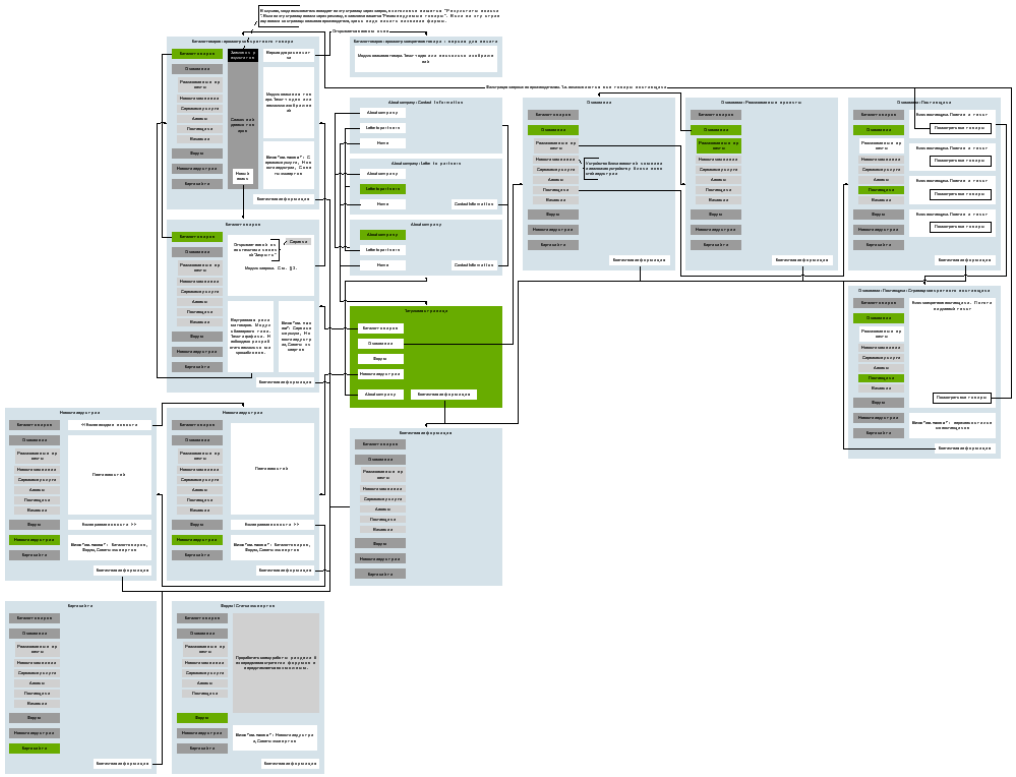

Рис.6-39. Итоговая модель графического интерфейса

Замечание по теме. Проектирование *от графического интерфейса* не касается архитектурных вопросов, т.е. выступает как замена бизнес- и системной аналитики.

Заключение. Что меня бесит

Некоторые «приятные мелочи» способны вывести из себя даже весьма флегматичного пользователя. В перечисленном списке кое-что является результатом незнание способов настройки приложений , кое-что представляет собой программные ошибки, кое-что является иллюстрацией несоблюдения перечисленных выше рекомендаций, кое-что появилось в результате реинжиниринга - не все пункты в старом меню попали в новые пиктограммы, кое-что унаследовано из старой архитектуры приложения:

- при установке курсора в поле ввода переключатель языка клавиатуры оказывается в положении, противоположном необходимому, Вы печатаете «по-английски» - ds gtxfnftnt gj fyukbqcrb;
- при запуске приложения-форматера по команде *печать* выводится запрос на обновление приложения;
- при манипуляциях с блоком текста меняется шрифт или форматирование соседних участков;
- для попадания на нужный элемент в линейке форматирования страницы необходимо точная подгонка курсора на несколько пикселов, элементы интерфейса расположенны оченьб близко;
- при выборе команд *download* предлагается установить специальное приложение;
- в MS Vilual Studio для продолжения редактирования необходимо остановить режим отладки специальной командой. В других IDE как-то без этого обходятся, новая компиляция и сборка приложения отменяет текущий режим отладки;

- В MS Word2010 в меню *Разметка страницы* имеется группа *Ориентация*, две кнопки которого меняют ориентацию **всего документа.** Для изменения ориентации **одной страницы** необходимо в том же меню выбрать в группе *Поля* пункт *Настраиваемые поля*, далее в форме с полным набором параметров для полей и ориентации выбрать пункт списка - *применить к выделенному фрагменту*.