3. Chapuis O., Roussel N. Metisse is not a 3D desktop. // ACM Press. October 2005

УДК 004.514.6
Гашко Р.В.
Научный руководитель: ***Костюк Д.А.***

ПРИМЕНЕНИЕ МОДЕЛИ ИНТЕРФЕЙСА КАЯ КРАУЗЕ ДЛЯ ОПТИМИЗАЦИИ ОФИСНЫХ ПРИЛОЖЕНИЙ

В работе рассмотрены проблемы организации интерфейса современных офисных программных пакетов на примере популярных текстовых процессоров. Предложен вариант оптимизации интерфейса с помощью применения модели Кая Краузе. Анализируются основные положения данной модели и их влияние на эффективность рассматриваемых приложений.

Среди проблем интерфейса современных офисных пакетов, исследователи обычно отмечают перегруженность пользовательского интерфейса (особенно это заметно в отношении инструментальных панелей), а также отсутствие решений, стимулирующих пользователя повышать свою квалификацию и, соответственно, выбирать более эффективные способы решения задач [1-2].

Современный текстовый процессор представляет собой сложный комплекс программ с огромным числом функций, каждую из которых разработчикам приходится делать доступной через графический интерфейс. Однако неоднократно проводившиеся наблюдения за пользовательской активностью показывают, что большая часть пользователей игнорирует такое многообразие возможностей, обходясь привычным минимумом, тем самым лишая себя возможности заметно облегчить и ускорить выполняемую работу. К сожалению, инерционность типичного пользователя отчасти поощряется главенствующим на рынке подходом, согласно которому менее используемые или требующие большей квалификации возможности приложения скрываются в менее заметных или более труднодоступных областях интерфейса. В результате приложение выглядит более простым и привлекательным, не уменьшая число возможностей программного продукта. Однако при этом не поощряется повышение пользовательской квалификации, и вероятность использования дополнительных возможностей приложения оказывается невысока.

В частности, в [2] анализируются приемы работы со стилями текста, диктуемые наиболее популярными текстовыми процессорами (рассмотрены Microsoft Word 2003 и OpenOffice Writer 2.0). Поддержка стилей реализована в любом современном текстовом процессоре. Стили позволяют пользователю один раз определить некоторый вариант форматирования текста, включая параметры шрифта, выравнивания, интервалов и др., а затем применять созданный стиль к различным блокам текста. Своим происхождением стили обязаны различным языкам разметки. Использование стилей в текстовом процессоре позволяет:

1. Упростить соблюдение единого стиля оформления документа,
2. Быстро изменить оформление документа, отредактировав стиль.

Как видно, выигрыш от использования стилей возрастает по мере увеличения объемов текста. Однако разработчики относят данный механизм к категории функций, предназначенных для квалифицированных пользователей. Как следствие, осмысленное применение стилей не является обязательным, а средства их редактирования зачастую тщательно укрыты от глаз неопытного пользователя (Microsoft Word, Abiword, KWord, Corel WordPerfect и др.). Поэтому большинством пользователей данный механизм игнорируется, в результате чего, во-первых, после набора объемного текста приведение его к нужной схеме оформления оказывается трудоемкой задачей, а во-вторых, оказывают

ся нефункциональными зависящие от использования стилей средства, такие как автоматическое генерирование оглавлений

Альтернативным является подход в котором информационное наполнение документа отделено от его оформления. В большей степени сюда можно отнести редакторы XML-кода, однако как минимум один текстовый процессор следует данной парадигме [3] По ряду причин эта модель не получила распространения при подготовке текстовых документов

Второй широко известной проблемой является перегруженность интерфейса Из-за графических средств доступа к большому числу возможностей, в интерфейсе полноценного текстового процессора значительную часть экрана занимают инструментальные панели с пиктограммами действий. Плотность размещения управляющих элементов на экране оказывается весьма высокой, а размер - как можно меньшим, что создает дополнительную нагрузку как на физическую (i), так и на умственную (ii) активность пользователя:

(i) Как известно, пользователь, совершающий физическое действие, не может сделать его одновременно быстрым и точным, а вынужден останавливаться либо на одном либо на другом варианте (в данном случае не рассматриваются вопросы особых физических данных или плоды специальной длительной тренировки). Главным образом это сказывается на использовании мыши, где важна как быстрота, так и точность движений [1, 2]. Для оценки оптимальности расположения управляемых мышью элементов интерфейса традиционно используется закон Фиттса, определяющий время t, необходимое на перемещение курсора мыши к элементу управления: $t = a + b\log_2(D/S + 1)$. Здесь S - размер объекта (точнее его ширина в направлении перемещения курсора), D - расстояние от начальной позиции курсора до объекта, а и b - весовые коэффициенты задающие параметры конкретного пользователя Время отсчитывается от момента когда курсор начинает движение по прямой линии, до момента, когда пользователь щелкает мышью по объекту. Таким образом, время t пропорционально расстоянию до объекта и обратно пропорционально размерам объекта (его ширине)

(ii) Чем большее количество вариантов выбора предоставлено пользователю тем больше времени ему потребуется, чтобы сделать выбор (Закон Хика) Выбор из одного списка эффективнее, чем из иерархии тех же вариантов, однако одноуровневые списки подвержены действию правила, известного как кошелек Миллера, согласно которому емкость кратковременной памяти человека ограничена пятью-девятью объектами (цифр, букв или названий предметов). Это целесообразно учитывать при группировке элементов интерфейса (кнопки панелей инструментов, пункты меню, вкладки, опции выбора и др.). Действительно, длинный ряд кнопок на панели инструментов обычно воспринимается гораздо хуже, чем те же самые кнопки, собранные по функциональному признаку в группы и разграниченные разделителем Многие исследователи советуют понижать максимально допустимое число элементов в группе до пяти, учитывая перегруженность современных интерфейсов информацией.

Интерфейсные модели, предлагаемые нами в качестве варианта решения очерченных проблем, были изначально реализованы в ряде программных продуктов для работы с растровой графикой. Автором как концепции интерфейса так и программных продуктов, ее реализующих является немецкий программист Кай Краузе (Kai Krause) при участии которого была организована софтверная фирма Meta Creations К числу наиболее известных продуктов последней относятся Kai's Power Tools (KPT), Kai's Power GOO, Kai's Photo Soap Данные программы привлекли значительное внимание пользователей, не в последнюю очередь благодаря тому, что весьма сложные алгоритмы были успешно скрыты в них за элегантными интерфейсными метафорами делавшими кривую обучения чрезвычайно пологой.

M Muller-Prove [6] в ходе анализа концепции интерфейса Кая Краузе выделяет следующие основные положения: метафора комнат (rooms metaphor), развертывание инст-

рументов (unfolding functionality), реакция на прохождение курсора мыши (mouse over), пятерка избранных (memory dots / five favorites), а также прозрачность и затенение (transparency & shadows)

Основная идея интерфейса – использование полноэкранного режима и разделение интерфейса на "комнаты", предназначенные для выполнения конкретного комплекса задач. В приложениях KPT элементы интерфейса маскируются таким образом, что – нет ни строки меню, ни окна изображения, ни рабочего окна. Ощущения пользователя схожи с пребыванием в комнате со специально созданным окружением для решения некоторой задачи. Одно из наиболее развитых приложений, Kai Photo Soap, в начале работы представляет собой серию из семи комнат - In, Prep, Tone, Color, Detail, Finishing и Out. Внимание пользователя полностью акцентируется на выполнении поставленной задачи, нет лишних инструментов. За счет этого разгружается интерфейс и увеличивается пространство под рабочую область. Недостатками этого приема является возможность работы с инструментами, актуальными только для активной

Развертывание инструментов подразумевает, что для каждого из комплексов решаемых задач активируются свойственные ему инструменты, а не используемые – затемняются (возможно также уменьшение размеров). Так, в KPT Convolver все задачи делятся на три комплекса: Explore, Design и Tweak. Основным преимуществом данного подхода является то, что внешний вид остается привычным. Все инструменты доступны сразу, т.к. если нажать на обесцвеченный инструмент, то автоматически осуществляется переключение на другой комплекс задач

Реакция на прохождение курсора мыши в настоящий момент используется в интерфейсах многих программных продуктов. Неактивные инструменты становятся невидимыми или полускрытыми, однако при подведении курсора мыши "оживают". Внимание пользователя больше фокусируется на задании, размытые предметы находятся вне поля зрения, за счет скрытия неактивных инструментов увеличивается рабочая область, пользователю проще ориентироваться в инструментах.

Пользователю предоставляется возможность выбора пяти наиболее часто используемых инструментов (или других элементов интерфейса) для быстрого переключения между ними. Избранные инструменты могут помещаться в специальную область экрана, а также могут быть доступны по клавиатурным сокращениям

Каждая пиктограмма в интерфейсе Кая отбрасывает тень, а диалоговые окна сделаны полупрозрачными. Несмотря на активные попытки применения эффектов прозрачности в современных графических интерфейсах (например выпадающие меню), целесообразность этого обычно невелика. Преимущество же данного приема в реализации Meta Creations заключается в том, что предмет, находящийся под полупрозрачной панелью остается видимым.

Кроме того, в приложениях Meta Creations наблюдается ярко выраженный объектно-ориентированный подход. Наиболее легко объектная парадигма и непосредственное управление применимы в САПР, где у отдельных объектов и инструментов имеются однозначные прототипы из реального мира, а также активно используется в программах создания схем и диаграмм, в редакторах векторной и растровой графики

При разработке объектно-ориентированного интерфейса актуальна проблема выбора удачных метафор и концепций. Эта задача тем сложнее, что удачную аналогию из реального мира можно подобрать отнюдь не для всех задач, причем факт ее существования или несуществования заранее неизвестен. Кроме того, аналогии часто накладывают дополнительные ограничения на объект: ограничения реального мира часто оказываются излишними в виртуальном и сохраняются исключительно для поддержания большего сходства. Сохранение же дополнительных функций у интерфейсного объекта нарушает метафору и требует дополнительного обучения пользователя. Так, далеко не сразу начинающий пользователь обнаруживает, что объекты можно перетаскивать не только в корзину, в папки и на

рабочий стол, но иногда также и на некоторые элементы управления, например кнопки. И даже сами папки не являются полной аналогией папок из внешнего мира - хотя бы потому, что могут содержать в себе другие папки, образующие иерархическую структуру вложенности. Очевидно из-за данного ограничения объектно-ориентированная парадигма в настоящий момент применяется в офисных приложениях весьма ограниченно.

В процессе исследования была выполнена адаптация модели интерфейса Кая Краузе для типичного текстового процессора.

В схеме диалога с текстовым процессором частично просматриваются аналогии с метафорой комнат. Так, в Microsoft Word помимо основного рабочего пространства можно заметить еще дополнительные: достаточно самостоятельное рабочее пространство «Предварительный просмотр», а также рабочие пространства с собственными меню и панелями инструментов, с которыми пользователь имеет дело при редактировании внедренных в документ объектов. Однако данная метафора находится в подчиненном положении, частично скрыта от пользователя и никак не стимулирует его к правильной очередности действий с документом.

В отличие от растровых изображений, с которыми работают продукты Meta Creations, текстовый документ имеет существенно более сложную иерархическую структуру. Внедренные объекты могут частично обладать свойствами содержащего их документа – например, ячейке таблицы или содержимому текстовой выноски свойственно форматирование текста, мало отличающееся по набору параметров от форматирования текста абзаца. В наиболее сложных вариантах вложенность объектов может быть многоуровневой (текстовая выноска, содержащая в себе таблицу или другую текстовую выноску). Поэтому в ряде случаев необходим автоматический переход между комнатами с сохранением позиции курсора и режима отображения документа.

При разбиении интерфейса на комнаты и выборе набора инструментов было проведено исследование пользовательской активности для выявления наиболее часто используемых функций интерфейса. Часть инструментов характерна только для своих собственных комнат, как в случае оригинальных графических приложений, однако для обеспечения автоматических переходов была предусмотрена категория инструментов присутствующих в нескольких комнатах сразу. Данные инструменты могут либо осуществлять при выборе переход в наиболее подходящую для их использования рабочую среду, либо действовать независимо во всех комнатах, в которых они видны. Для реализации метафоры комнат были выбраны следующие рабочие режимы, отвечающие различным этапам работы:

- открытие документа (включает список последних использовавшихся документов в виде альбома, а также кнопку вызова диалога открытия файлов);
- параметры документа (параметры страницы и средства редактирования стилей);
- редактирование документа (сюда относятся инструменты для манипулирования параметрами шрифта и абзаца, выбора стиля, работы с буфером обмена, поиска и автозамены, отмены и повтора действий пользователя, а также перехода по вставке объекта либо таблицы);
- вставка (средства вставки полей, символов, переход к комнате работы с таблицами, переход к комнатам, отвечающим за работу с внедренными объектами (формула, фигурный текст, рисунок и др.);
- работа с web-документами (вставка и редактирование средств разметки гипертекста),
- работа с таблицами;
- ряд комнат для работы с внедренными объектами;
- слияние документов (подготовка документов на основе бланков и источника данных);
- сохранение документа (включая также окончательный просмотр, печать и рецензирование).

ЛИТЕРАТУРА:

1. Д. Раскин. Интерфейс: новые направления в проектировании компьютерных систем. СПб.: Символ-Плюс, 2003 – 272 с.

2. B. Byfield. OpenOffice.org Writer vs. Microsoft Word // NewsForge: the online newspaper for Linux and Open Source. 12.06.2005.
http://software.newsforge.com/article.pl?sid=05/06/14/2137222&from=rss
3. WYSIWYM. The WYSIWYM paradigm in software engineering. 21.06.2006.
http://en.wikipedia.org/wiki/WYSIWYM
4. M. Müller-Prove. The Interface of Kai Krause's Software. ASI Software-Ergonomie Lehre Seminare. University of Hamburg. 1999. http://www.mprove.de/script/99/kai/index.html

УДК 519.876.5+004.514.6:657.1
Войцехович Л. Ю.
Научный руководитель: д.т.н., профессор Головко В. А.

НЕЙРОСЕТЕВЫЕ МЕТОДЫ ОБНАРУЖЕНИЯ АТАК НА КОМПЬЮТЕРНЫЕ СЕТИ

1. ВВЕДЕНИЕ

Одной из форм глобализации мирового пространства является информационная глобализация, которая связана с широким распространением сети Интернет. Информационная глобализация увеличивает степень уязвимости компьютерных систем, что уменьшает их безопасность. *Атакой на компьютерные сети* называется совокупность определенных действий, приводящих к подрыву безопасности системы. В результате атаки злоумышленник может получить доступ к конфиденциальной информации или нарушить нормальное функционирование системы. Это приводит к большим материальным и социальным издержкам.

Важным этапом обеспечения безопасности компьютерных систем является проектирование *систем обнаружения атак* (Intrusion Detection System – IDS). Такие системы способны на основе анализа сетевого трафика автоматически обнаруживать атаки TCP/IP, что позволяет предпринять необходимые меры для нейтрализации угрозы.

В данной работе рассматриваются нейросетевые подходы для построения систем обнаружения атак. В качестве базы данных для тестирования системы используется KDD-99 [1], которая содержит почти 5 миллионов записей соединений и 41 параметр сетевого трафика. При этом атаки делятся на четыре основных класса: DoS, U2R, R2L и Probe.

Атака DoS – отказ в обслуживании, характеризуется генерацией большого объема трафика, что приводит к перегрузке и блокированию сервера.

Атака U2R предполагает получение зарегистрированным пользователем привилегий локального суперпользователя (администратора).

Атака R2L характеризуется получением доступа незарегистрированного пользователя к компьютеру со стороны удаленной машины.

Атака Probe заключается в сканировании портов с целью получения конфиденциальной информации.

В работе предлагаются различные варианты построения систем обнаружения атак, которые базируются на использовании рециркуляционных и многослойных нейронных сетей. Результаты экспериментов обсуждаются.

2. ГЕНЕРИРОВАНИЕ АРХИТЕКТУРНЫХ РЕШЕНИЙ

Рассмотрим различные архитектурные решения для построения систем обнаружения атак. В качестве входных данных используется 41-размерный вектор, который характеризует параметры соединения сети. Задачей IDS является обнаружение и распознавание атак. Поэтому в качестве выходных данных используется m-мерный вектор, где m равняется количеству атак плюс нормальное состояние.