СИСТЕМА АДАПТАЦИИ ПОЛЬЗОВАТЕЛЬСКОГО ИНТЕРФЕЙСА

УДК 621.391.246

ХОДАКОВ Виктор Егорович
д.т.н., профессор, заведующий кафедрой Информационных технологий Херсонского национального технического университета.
Научные интересы: автоматизированные системы управления, современные информационные технологии.

ВЕЛИЧКО Юрий Игоревич
ассистент кафедры Информационных технологий Херсонского национального технического университета.
Научные интересы: адаптивные пользовательские интерфейсы.

ВВЕДЕНИЕ

Задачи повышения уровня взаимодействия в системе «человек-компьютер» поставлены перед разработчиками информационных систем (ИС) с первых ЭВМ и не теряют своей актуальности в наши дни. Для повышения качества взаимодействия за годы развитии компьютерной техники были выработаны походы и рекомендации по проектированию эргономичных человеко-машинных интерфейсов, основанные на результатах исследований в когнитивной и инженерной психологии. Сегодня разрабатываются программные продукты (ПП) с эргономичными пользовательскими интерфейсами (ПИ). Но даже самые качественные ПИ, удобные для одних групп пользователей, могут быть непонятными для целого ряда других в связи индивидуального восприятия и специфичностью навыков каждого человека. Что является удобным для одних пользователей может быть не понятным, либо неочевидным, для других[1,2]. Решением данной проблемы может быть включение в интерфейс аппарата изменяющего состояние интерфейса согласно нужд пользователя – системы адаптации интерфейса (САИ).

Целью работы является изложение универсальной системы адаптации пользовательского интерфейса. В статье будет кратко описана модель, структура и алгоритм функционирования САИ. Для формирования понимания сущности САИ необходимо учесть следящее:

1. Параметры графического пользовательского интерфейса – структура и состав ПИ ПП.
2. Пользовательский интерфейс как объект управления, математической модель описывающая процесс управления интерфейсом.
3. Модель пользователя – информационная модели, агрегирующая показатели характеристик пользователя.
4. Место САИ в структуре программного продукта – определение структуры САИ и способа его внедрения в программный продукт.
5. Структура клиентской части САИ – архитектура и алгоритмы работы клиентской части САИ.
6. Структура ядра САИ - архитектура и алгоритмы работы ядра САИ.

ОСНОВНАЯ ЧАСТЬ

Параметры интерфейса. Графический пользовательский интерфейс (ГПИ) любого программного продукта, состоит из небольших «строительных блоков», называемых элементами либо компонентами интерфейса (ЭИ), образно говоря, представляет собой структуру наподобие детского конструктора. К элементам интерфейса относят: кнопки, меню, пиктограммы, поля ввода текста и прочие компоненты. Однако не все элементы ГПИ являются видимыми для пользователя, например компоновщики (layout), выполняют функции размещения элементов на определенном участке, и

объединения в функциональные группы. На первый взгляд ЭИ различаются по функциональному назначению, внешнему виду и способу компоновки. Однако, все ЭИ, особенно видимые, обладают множеством одинаковых свойств – цвет фона, размер шрифта, цвет шрифта, расширяемость, привязка, пиктограмма, интерактивная подсказка и т.д. Значения этих свойств, а так же порядок размещения ЭИ определяет общий вид интерфейса.

Интерфейс ПП, своим внешним видом, формирует у пользователя внутреннею модель системы. Чем «прозрачнее» [3,4] и понятнее он будет тем эффективнее пользователь может решать свои задачи. Компоновка и внешний вид элементов интерфейса определяется дизайнерским замыслом и эргономическими законами. Группировка и сортировка компонентов, последовательность расположения, открытость и вложенность, взаимосвязь и последовательность обработки – все это влияет на когнитивные аспекты работы пользователя. Параметры элементов могут зависеть от многих факторов. В программах и приложениях вид ЭИ в первую очередь определяется параметрами интерфейса операционной системы, поскольку разработчики, при реализации интерфейса используют «системные» элементы. В вэб-приложениях не существует столь жестких технических ограничений, здесь основным фактором является дизайнерская задумка, именно в ней определяется внешний вид и значения свойств элементов.

Особенностью создания ПП на сегодняшний день, является необходимость реализации на нескольких, совершенно независимых платформах. При этом функциональная часть системы, так называемая бизнес-логика, остается неизменной – меняется только интерфейс системы. Это еще больше усложняет процесс взаимодействия человека с компьютером, поскольку пользователь должен приспосабливаться уже не к одному, а к нескольким интерфейсам одной и той же системы. Задача проектировщика интерфейса, становится еще сложнее. Теперь он должен учитывать особенности отображения компонентов интерфейса в разных средах. Вопрос об адаптации интерфейса под нужды конкретного пользователя, в таком случае, отходит на последнее место из-за большой сложности реализации. Проблемы взаимодействия решаются применением эргономических требований проектирования интерфейса, ориентированных на среднестатистического пользователя.

Рассмотрим зависимость вида интерфейса ПП от внешних факторов (рис. 1)

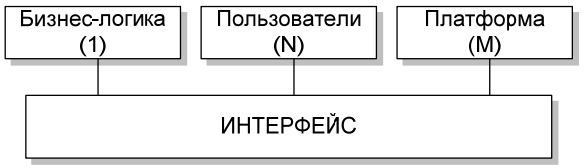

Рисунок 1 – Факторы, влияющие на внешний вид интерфейса ПП

Поскольку бизнес-логика (back-end) любого программного продукта представляет собой реализацию системы решающую определенную задачу то и интерфейс (front-end) этой программы должен отражать только модель этой системы и нечего более. Количество пользователей системы, практически неограниченно. То же самое можно сказать и про платформу функционирования (реализацию) – даже если в текущий момент система разработана для работы в одной среде, то со временем возможен ее перенос на другую платформу. При этом интерфейс должен оставаться понятным пользователю и способствовать продуктивной работе. Таким образом реализация интерфейса жестко ограничена моделью системы, должна соответствовать требованиям предъявляемой той или иной платформой и быть понятной для любого пользователя. Для достижения этой цели предлагается внедрить модуль, которой бы управлял состоянием интерфейса и адаптировал его под нужды текущего пользователя.

ИНТЕРФЕЙС КАК ОБЪЕКТ УПРАВЛЕНИЯ

В процессе адаптации интерфейс выступает как объект управления. Если состоянием интерфейса управляет человек-оператор, то этот процесс стоит считать настройкой интерфейса. Если параметрами интерфейса управляет отдельный модуль ПП, основываясь на характеристиках и действиях человека оператора (ЧО) – этот процесс можно назвать адаптацией интерфейса к пользователю. Поскольку адаптацию можно представить как управление, то под управлением будем понимать процесс организации такого целенаправленного воздействия на интерфейс, в результате

которого он переводится в требуемое (целевое) состояние. Объектом управления будем называть ту часть интерфейса, состояние которой нас интересует в текущей ситуации и на которую можно целенаправленно воздействовать — управлять. В рамах вопроса улучшения взаимодействия в системе «человек-машина» объектом управления является интерфейс системы в целом. В самом процессе адаптации – объектами управления будут являться компоненты интерфейса. Субъектом управления будет выступать человек оператор, поскольку адаптация направлена на удовлетворение его потребностей [3].

Обозначим через Y – состояние интерфейса формирующего информационную модель объекта, и через Z^* – представление пользователя о состоянии системы – концептуальную модель. Тогда равенство $Y = Z^*$ – означает, что интерфейс отражает состояние системы в понятном и наиболее «удобном» виде. Если $Y \neq Z^*$ – это означает, что пользователь не может реализовать свои цели посредством текущего интерфейса, либо внешний вид интерфейса не располагает к удобной работе. Соответственно необходимо ввести модуль приводящий интерфейс в необходимое состояние по средством управления интерфейсом.

Для реализации управления необходим канал управления U , с помощью которого можно влиять на состояние интерфейса:

$$Y = F^0(X, U)$$

где X – воздействующие на интерфейс факторы, F^0 – оператор преобразования интерфейса, учитывающий наличие фактора управления U .

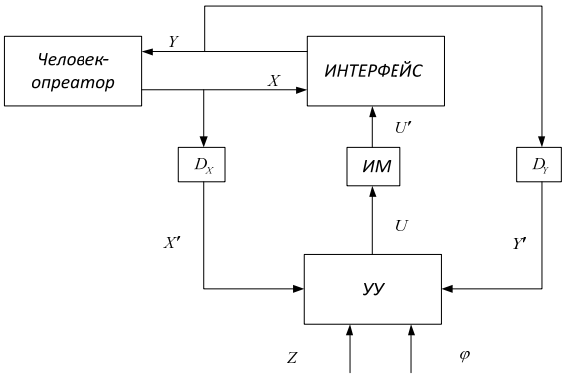

Рисунок 2 – Блок-схема системы управления интерфейсом

Представим систему реализующую совокупность алгоритмов обработки информации для управления интерфейсом. Система управления анализирует параметры человека-оператора и его действия, изменяющих интерфейс. Блок-схема системы управления интерфейсом приведена на рис. 2.

Здесь D_X – датчик, измеряющий состояние пользователя и регистрирующий его действия, D_Y – датчик, измеряющий состояние интерфейса. Результаты измерений $X' = D_Y(X); Y' = D_Y(Y)$ поступают на управляющее устройство (УУ), вырабатывающие команды управления *U*. Команды управления обрабатываются исполнительными механизмами (ИМ), для изменения состояния управляемого входа U' интерфейса. Управляющее устройство формирует управляющее воздействие:

$$U = \varphi(X', Y', Z*)$$

Где Z^* – цель управления; φ – указание, как добиваться поставленной цели.

Управление связано прежде всего с целями $\{Z^*\}$, которые поступают в систему управления. Эти цели формирует человек-оператор, который и является потребителем будущей системы управления объектом. ЧО выступает в качестве заказчика и потребителя создаваемой системы управления.

В реальных системах ЧО не всегда может самостоятельно задавать корректные цели. Это обусловлено двумя факторами: во-первых, цель, поставленная ЧО, не всегда может быть реализована системой, и во вторых ЧО может не знать о возможности постановки цели (изменения интерфейса). В таких случаях функцию постановки цели должен брать на себя сам интерфейс, а точнее его управляющая или адаптивная часть.

МОДЕЛЬ ПОЛЬЗОВАТЕЛЯ В ИНТЕРФЕЙСЕ

Ключевой фигурой в этом процессе является человек, который формирует цели управления $\{Z^*\}$. Действия ЧО влияют на УУ, что ведет за собой изменение параметров интерфейса. Управление состоянием интерфейса должно осуществляться на основании информации полученной от пользователя на протяжении всех сеансов взаимодействия. Для этого УУ должно иметь возможность запоминания и хранения данных. УУ должно иметь возможность осуществлять управле-

ние с учетом, формального представления о пользователе – модели пользователя (МП. Чем адекватнее будет эта модель, тем точнее будет осуществляться управление.

Одним из подходов реализации модели пользователя является реализации в виде множества профессиональных, психологических и физиологических характеристик. МП в таком случае представляется как абстрактная структура данных – список элементов, где каждый элемент представляет пару «свойство-значение» [4, 5]:

$$Wm(t) = \begin{cases} W_{type}, \\ (profact(i), value\ \ profact(i)), \\ \dots \\ (psyfact(j), value\ \ psyfact(j)). \end{cases}$$

где W_{type} – наименование одного из обобщенных типов пользователя, которое зависит от реализации системы и области ее применения; $profact(i)$ – наименование профессиональной i-ой характеристики пользователя; $psyfact(j)$ – наименование j-ой психологической характеристики пользователя; $value\dots$ – значения соответствующего показателя.

Деятельность пользователя может быть описана следующим образом. Пользователь на основании взаимодействия с интерфейсом формирует индивидуальный информационный образ некоторой ситуации и реализует поведение, т.е. свою деятельность в данной ситуации [3]:

$$W(t) = [\mathrm{Im}, Pm, \mathrm{Im}(m)]$$

где Pm – индивидуальный оператор поведения; $\mathrm{Im}(m)$ - информационный образ ситуации.

$$\mathrm{Im} = [\mathrm{Im}, Q, t, \tau]$$

где Q множество внешних взаимодействий на сенсорную систему пользователя; t – текущий момент времени; τ – время запаздывания обусловленное инерционностью сенсорной системы.

Применение такого подхода дает возможность сформировать модель пользователя для системы адаптации интерфейса компьютерной системы. Важно отметить, что часть параметров модели отражающие личностные характеристики пользователя – не зависит от предметной области. Другая часть – отражающая профессиональные качества, непосредственно зависит от области функционирования системы.

Создание МП на основании характеристик пользователя является наиболее практичным подходом за счет детально формализованной и предметно независимой структуры. Но применение МП подобной конфигурации, в системе адаптации интерфейса, все еще не обеспечивает адекватное отображение реального пользователя. Адаптация интерфейса основанная на МП построенной по принципу ключ-значение, где ключом служит имя ХП, а значением, соответственно значение ХП, не будет столь гибкой как того хотелось бы, из за жестких условий, согласно которым изменение интерфейса произойдет только в том случае если значение ХП в модели определено и этому значению сопоставлен некий сценарий адаптации.

Реализация процесса адаптации для каждого пользователя является достаточно сложной в силу большого количества характеристик пользователей и большого диапазона значений, а так же возможных комбинаций характеристик. Наиболее оптимальным решением является разбиение пользователей на классы как это делается при стереотипном подходе. Для этого исходное множество возможных пользователей U разобьем на нечеткие подмножества $U = \left\{A_i^j\right\}, i = \overline{1,n}, j = \overline{1,m}$, где n – количество характеристик пользователя от которых зависит адаптация интерфейса; j – количество значимых величин i-ой ХП. В этом случае при адаптации учитывается не все множество значений ХП, а $k = \sum_{i=1}^{n}\sum_{j=1}^{m} A_j^i$ показателей, представляющих определенные подмножества пользователей. Будем называть подмножество пользователей A_j^i – группой пользователей (ГП). Разбиение пользователей на группы позволяет увеличить гибкость процесса адаптации.

В этом случае модель пользователя может быть представлена в виде:

$$Wm(t) = \{group(A_i), value\ group(A_i)\},\ i = \overline{1,n}$$

где $group(A_i)$ – наименование i-ой группы пользователя; $value\ group(A_i)$ - значения определяющее принадлежность текущего пользователя к i-ой группе; n - количество всех групп пользователей в

системе. Данный подход решает проблему обобщения характеристик пользователя, имеющих большой диапазон возможных значений. Характеристическая функция причастности пользователя к A_i группе на основании теории нечетких множеств будет иметь вид[7]:

$$\chi_{A_i} : X \to \{0.1\}$$

Тогда МП можно представить в табличном виде (табл. 1).

Таблица 1 –

Модель пользователя при стереотипном подходе

	A_0	A_1	…	A_n
МП	1	0	…	1

Однако такое решение все же является недостаточно гибким, и не может предоставлять альтернативные пути адаптации потому, что подобная МП дает жесткое определение принадлежности пользователя к ГП, в соответствии с функцией принадлежности МП к ГП:

$$\mu_{A_i}(x) = \begin{cases} 1, & \text{если } x \in A_i, \\ 0, & \text{если } x \notin A_i. \end{cases}$$

Что бы обойти это ограничение было решено использовать аппарат нечеткой логики. Для синтеза МП, основанной на показателях принадлежности к той или иной ГП. Примем за X множество возможных значений ХП, $X = \{x_i\} i = \overline{1,m}$ где m - количество возможных значений ХП. Тогда каждой ГП может быть поставлено в соответствие нечеткое множество пользователей A, такое, что для любого x выполняется условие[6]

$$x \in X, \mu_A(x) \in [0,1]$$

В этом случае значение $\mu_A(x)$ будет показывать степень принадлежности МП к A -ой группе пользователей. Тогда модель пользователя можно представить в виде, показанном на табл. 2.

Таблица 2 –

Модель пользователя при стереотипном подходе на основе нечеткой логике

	A_0	A_1	…	A_n
МП	0,75	0,3	…	0,0

Такая модель более точно отражает характеристики пользователя, за счет чего позволяет системе адаптации производить изменение интерфейса на более приближенной к реальному пользователю модели. Данная модель позволяет увеличить гибкость адаптации за счет предоставления возможности ранжирования сценариев адаптации в зависимости от показателей принадлежности текущей МП к определенной ГП.

МЕСТО САИ В СИСТЕМЕ ЧЕЛОВЕК-МАШИНА

Разработчики ПО наработали целый ряд конструкций, применение которых способствует увеличению производительности труда программиста и улучшению структуры программы за счет универсальных подходов по решению аналогичных задач. Эти конструкции получили название – «шаблоны проектирования» или «паттерны проектирования». Если программное обеспечение разрабатывается без применений каких либо паттернов, то его структура будет не такой гибкой, и носит хаотичный характер. Увеличиваются затраты на доработку и улучшения. Условия нынешнего уровня развития ИТ и рынка ПП, практически не позволяют реализовывать ИС такими подходами. Поэтому разработка всех качественных программных продуктов ведется с применением тех или иных шаблонов проектирования[8]. Разработка программ с пользовательским интерфейсом, как правило, происходит согласно одному из следующих паттернов:

- Facade – Фасад;
- Model-View-Controller(MVC) – Модель-Представление-Контролер;
- Model-View-Presenter(MVP) – Модель-Представление-Представитель;
- Model-View-ViewModel(MVVM) – Модель-Представление-ПредставлениеМодели.

Разумеется, что существуют и другие принципы построения программ с пользовательским интерфейсом, но их зачастую можно свести к вышеперечисленным, так как они являются их порождением.

Основной чертой вышеуказанных методов является отделение бизнес-логики системы от ее интерфейса, что позволяет говорить о проектировании, реализации и последующем функционировании интерфейса как об отдельно стоящих задачах, ни как не связанных с логикой работы программы. По этому если ПП разраба-

тывается по одному из выше перечисленный паттернов, то модуль адаптации может быть внедрен в его интерфейс без какого либо вмешательства в структуру и в процессе функционирования системы. САИ ни каким образом не зависит от предметной области или реализации ядра программного продукта, поскольку ее единственной задачей является изменение интерфейса согласно требований текущего пользователя. Тогда место модуля адаптации, определяется его функциональным назначением и требованиями к реализации, как это показано, на рис. 3.

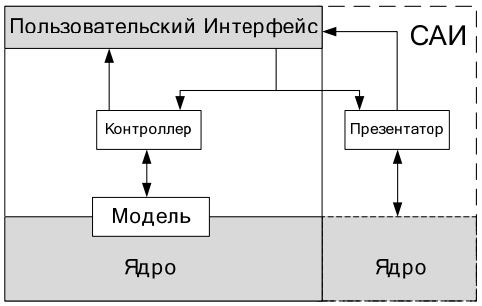

Рисунок 3 – Структура программного продукта с САИ

САИ может быть построена по шаблону MVP, в котором презентатор – структура, отвечающая за взаимодействие между интерфейсом и ядром САИ. По сути, презентатор выполняет две функции:

- Регистрацию событий интерфейса и осведомление ядра САИ об их возникновении.
- Изменение параметров интерфейса, согласно инструкциям выработанных ядром САИ.

Ядро САИ – представляет собой программную реализацию адаптации пользовательского интерфейса. Основной задачей ядра САИ является изменение параметров интерфейса пользователя в зависимости от текущей МП.

Основное преимущество построения САИ по выше представленной модели является отделение логики адаптации интерфейса от механизмов реализации этой адаптации. Ядро САИ является независимым модулем, не связанным с интерфейсом напрямую, что приводит к тому, что ядро САИ может быть реализовано независимо от интерфейса. В таком случае функции изменения интерфейса возлагаются на контроллеры (презентаторы) САИ. Они реализуют команды изменения интерфейса в терминах самого интерфейса. Контроллер САИ зависит от способа реализации интерфейса и должен быть с ним полностью совместим.

Такой подход построения САИ позволяет реализовать адаптацию по клиент-серверной схеме, когда одно ядро производит адаптацию нескольких интерфейсов совершенно различных по реализации программных продуктов (рис. 4).

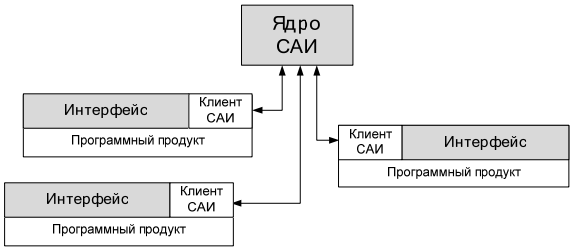

Рисунок 4 – Клиент серверная архитектура, включающая САИ

Возможность реализации клиент-серверной архитектуры позволяет использовать САИ не только в конкретном продукте, а одновременно в нескольких различный приложениях, работающих с одной и той же системой. При этом адаптация интерфейса производится синхронно для конкретного пользователя на всех устройствах, с помощью которых он взаимодействует с системой.

СТРУКТУРА КЛИЕНТСКОЙ ЧАСТИ САИ

Клиентская часть САИ реализует изменение состояние интерфейса, согласно сценариям полученным от ядра. В ее задачи входит:

1. Регистрация действий пользователя.
2. Формирования пакета сообщений о действиях пользователя.
3. Оповещение ядра о действиях пользователя.
4. Наблюдение за ядром САИ на предмет возникновения управляющих воздействий.
5. Интерпретация сценариев адаптации.
6. Реализация адаптации.

В соответствии и выполняемыми задачами структура САИ имеет следующий вид (рис. 5):

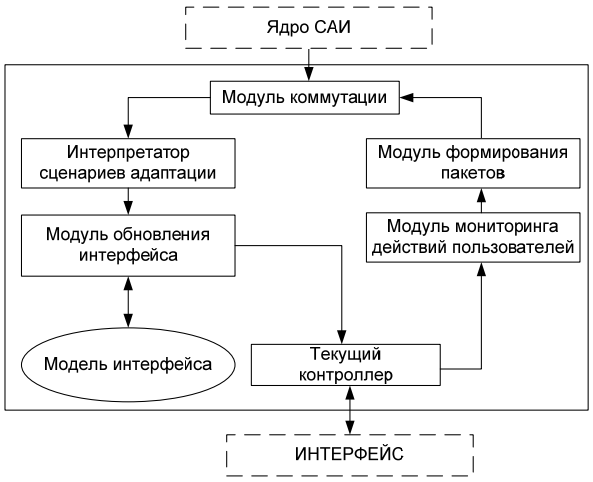

Рисунок 5 – Структура клиентской части САИ

Рассмотрим составляющие компоненты клиентской части САИ.

Модуль коммутации обеспечивает взаимодействие между клиентом и ядром. В его функции входит отправка и прием пакетов, мониторинг канала связи и оповещение об инициации процесса адаптации.

Интерпретатор сценариев адаптации преобразовывает полученный сценарий в последовательность команд по изменению интерфейса. Сценарий имеет декларативный характер, и состоит из набора команд имеющих вид:

«Изменить размер шрифта всех кнопок на X *»,*

где X – величина определяемая ядром САИ на основе текущей МП. Сценарий адаптации может состоять из нескольких подобных шагов для разных компонентов интерфейса.

Модуль обновления интерфейса выполняет функции изменения модели интерфейса и управления текущим контроллером. В соответствии с полученным набором команд модуль изменяет вид интерфейса через текущий контроллер и записывает полученные изменения в модель интерфейса. Это делается для того, что бы при смене контроллера, была возможность использования команд по адаптации полученных на предыдущих шагах.

Модель интерфейса представляет собой набор характеристик интерфейса.

Модуль мониторинга действий пользователя отслеживает работу пользователя с системой. В этот модуль должны быть заложены шаблоны поведения пользователей. При выявления соответствия между действиями пользователя и определенным шаблоном, модуль формирует сообщение и передает его на отправку ядру САИ.

Модуль формирования пакетов структурирует данные, полученные от модуля мониторинга действий пользователя, и выполняет отправку данных ядру САИ. Модуль необходим для того, что бы САИ не проводила адаптацию «мгновенно», отвечая каждому изменению в МП. Более логичным является накопление таких сообщений в пределах одного сеанса работы или определенного отрезка времени и пересылка их ядру. После чего эти данные будут обрабатываться в виде неделимой операции и в результате будет получен ряд команд по изменению состояния интерфейса.

СТРУКТУРА ЯДРА САИ

Ядро САИ состоит из двух основных компонентов – базы знаний системы адаптации интерфейса (БЗСАИ), которая хранит и обрабатывает данные на основе которых производится адаптация, и модуля реализующего алгоритмы адаптации. Общая структура ядра САИ представлена на рис. 6.

В БЗСАИ присутствуют статические и динамические данные. К статическим относится только база данных компонентов интерфейса (БДКИ). Эта структура содержит перечень всех возможных элементов интерфейса, их свойств и возможных значений этих свойств. БДКИ не зависит от реализации интерфейса. Пример структуры и реализации БДКИ можно найти в [11].

Динамические данные изменяются в процессе жизненного цикла САИ. Они могут быть заданы на этапе конфигурирования системы под конкретную предметную область, либо добавлены в процессе работы. Ниже перечислены динамические компоненты БЗСАИ.

База знаний о характеристиках пользователя (БЗХП). Эти данные формируются на этапе конфигурирования системы. БЗХП состоит из двух основных частей – базы данных характеристик пользователя (БДХП) и базы данный групп пользователей (БДГП). БДХП – отражает перечень характеристик пользователя от которых зависит адаптация интерфейса. К характеристикам пользователя относятся, например, пол, возраст, уровень компьютерной грамотности, скорость набора текста, предпочтительная цветовая схема, те-

кущее психологическое состояние и многие другие. БДГП – отражает перечень групп пользователей с соответствующими характеристиками пользователей. Например, для характеристики пользователя «возраст» может существовать пять групп пользователей – «дети», «подростки», «взрослые», «пожилые» и «возраст не определен».

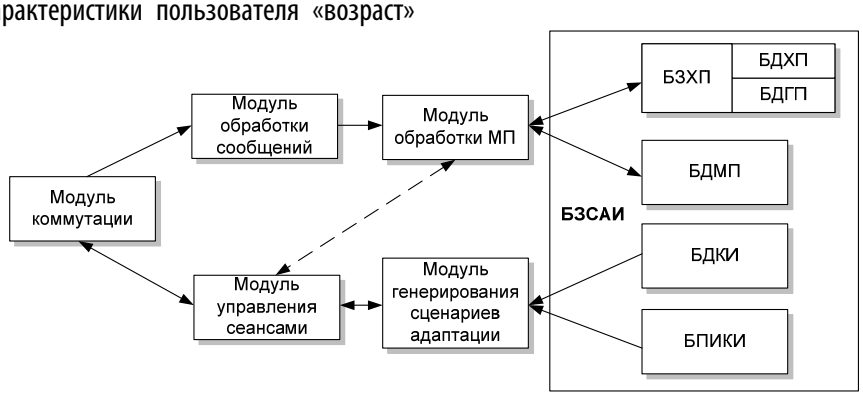

Рисунок 6 – Структура ядра САИ

База данных моделей пользователей (БДМП) формируется в процессе работы системы и постоянно изменяются. БДМП включает в себя модели всех зарегистрированных пользователей системы. В начале работы ядро САИ инициализирует сеанс и ассоциирует его с одной из МП. При добавлении нового пользователя в БДМП добавляется новая запись с МП.

База правил изменений компонентов интерфейса (БПИКИ) – эти данные формируются на этапе конфигурирования системы после того как определены все параметры МП – то есть группы пользователей, на которые будет ориентироваться адаптация. Каждое правило определяет зависимость состояния элементов интерфейса от МП, и включает методы изменения свойств компонентов.

Модуль коммутации – это механизм обеспечивающий взаимодействие между клиентами и ядром САИ. Его основной задачей является диспетчеризация сообщений.

Модуль обработки сообщений выполняет функции разбора полученных от клиента сообщений и преобразование их в последовательность действий по изменению МП.

Модуль обработки МП производит необходимые расчеты, и изменяет МП.

Модуль управления сеансами выполняет функцию структуризации клиентских подключений и выделения системных ресурсов для реализации адаптации.

Модуль генерирования сценариев адаптации создает, на основании измененных параметров МП сценарии, которые должны выполнить клиенты САИ по изменению состояния интерфейса.

Алгоритм функционирования ядра САИ состоит из следующих шагов (рис. 7).

Ожидание событий. В этой фазе ядро САИ находится в ждущем режиме и не предпринимает никаких действий. Событием, активизирующим работу, считается поступления сообщений – проявление активности со стороны клиентов.

Инициализация сеанса. Сеанс представляет собой некоторое пространство ресурсов, ассоциируемое с определенным пользователем. Инициализацией является загрузка необходимых данных о пользователе (МП). К одному сеансу может быть подключено одновременно несколько клиентов обеспечивающих адаптацию интерфейса для одного и итого же пользователя. Активация сеанса – представляет собой процесс предоставления сеансу ресурсов системы для выполнения определенных действий.

Обработка сообщений. На этой стадии ядро анализирует полученные от клиента данные о действиях пользователя. Если сообщения приходят от клиента, для пользователя которого не был установлен сеанс, то управление переходит на этап инициализации сеанса. Если сеанс уже существует, то яро, согласно полученному сообщению формирует, команды по изменению МП для текущего сеанса.

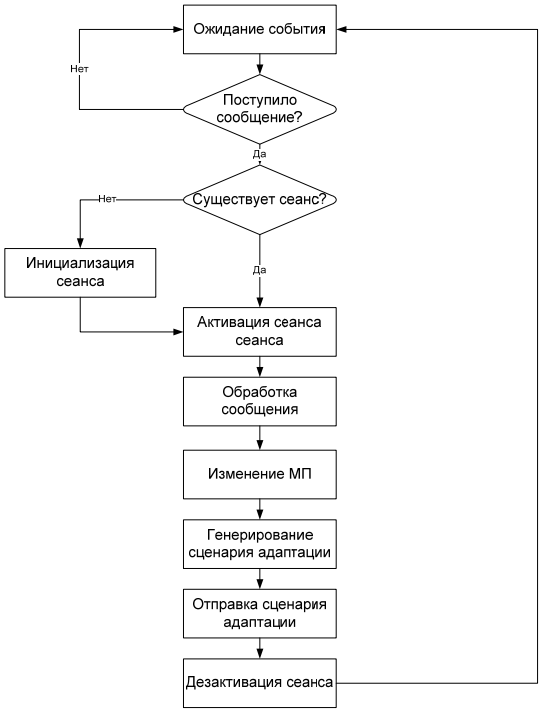

Рисунок 7 – Обобщенный алгоритм функционирования ядра САИ

Изменение текущей модели пользователя. На этой фазе ядро САИ производит корректировку значений в текущей МП, на основе полученных от клиента данных.

Генерация сценария адаптации. На этой фазе ядро САИ генерирует сценарий адаптации, согласно изменениям в МП, произведенным на предыдущем шаге.

Отправка сценария. После того как сценарий адаптации получен – ядро рассылает его всем клиентам текущего сеанса через модуль коммутации.

Дезактивация сеанса. Этот процесс представляет собой освобождение ресурсов системы и помещения сеанса в очередь ожидания.

ЗАКЛЮЧЕНИЕ

В данной работе дано общее описание сущности и структуры универсальной системы адаптации пользовательского интерфейса, которая может быть использована в программных продуктах практически любой направленности. В САИ интерфейс рассматривается как сложный объект управления. Ключевым звеном процесса адаптации считается нечеткая модель пользователя, отражающая характеристики человека-оператора. САИ, на основании текущей МП формирует сценарии адаптации, которые реализуются непосредственно в самом интерфейсе ПП. Использование широко распространенных шаблонов проектирования позволяет встраивать САИ в ПП не касаясь реализации системы, а так же строить САИ по клиент-серверной технологии. Использование САИ позволит создавать информационные системы с более дружелюбными ПИ и избавит от необходимости повторной реализации механизмов адаптации при переносе системы на разные платформы.

ЛИТЕРАТУРА:

5. Ломов Б.Ф. Человек и техника. – М: «Советское радио», 1966. – 464 с.
6. Литвак И.И., Ломов Б.Ф., Соловейчик И.Е. Основы построения аппаратуры отображения в автоматизированных системах. – М: «Советское радио», 1957. – 352 с.
7. Alan Cooper. About Face 3: The Essentials of Interaction Design /Alan Cooper, Robert Reimaann, Dave Cronin – Willey Publishing, 2007. – 651 p.
8. Мандел Тео. Разработка пользовательского интерфейса. – М.: ДМК Пресс, 2001.
9. Растригин Л.А. Адаптация сложных систем. – Рига: Зинатне, 1981. – 375 с.
10. Радванская Л.Н., Ходаков Д.В. Модели деятельности и адаптивный интерфейс //Вестник Херсонского государственного технического университета. – 1999. – №5. – С.12-17.
11. Кирхар Н.В., Ходаков Д.В. Модели деятельности пользователя компьютеризированной системы. //Вестник Херсонского национального технического университета. – Информационные технологии. – 2007. – №4 (27). – С.370-378.
12. Тэрано Т. Прикладные нечеткие системы: Пер. с япон. /К. Асаи, Д. Ватада, С. Иван и др.; под редакцией Т. Тэрано, К. Асаи, М. Сугэно. – М.: Мир, 1993. – 368 с.
13. Яхъяева Г.Э. Нечеткие множества и нейронные сети. Серия: Основы информационных технологий. – М: БИНОМ, 2008. – 320 с.
14. Fauler M. Patterns of Enterprise Application Architecture – Addison Wesley, 2002. – 506 p.
15. Ходаков В.Е., Величко Ю.И. Компоненты адаптации пользовательского интерфейса //Вестник Херсонского национального технического университета. – 2011. – №2 (41). – С.276-283.