Ontology-Based Design of Intelligent Systems User Interface

Boriskin A.S., Koronchik D.N.
Zhukau I.I., Sadouski M.E.
Belarusian State University of Informatics
and Radioelectronics,
Minsk, Belarus
Email: coloss_000@mail.ru
Email: denis.koronchik@gmail.com

Khusainov A.F.
Institute of Applied Semiotics of the
Tatarstan Academy of Sciences
Kazan, Russia
Email: Khusainov.aidar@gmail.com

***Abstract*—The work is devoted to the development of intelligent systems user interfaces design technology, which is based on an ontological model of the interfaces itself and the ontological model of the design process.**

***Keywords*—user interface, subject domain, ontology, intelligent system, ontology-based design, user, interface action, interface command, message.**

I. Introduction

A. Objective and Relevance of the work

The objective of this paper is the creation of technology of design of multimodal intelligent systems user interface and the creation of visualization tools for intelligent systems in particular. The relevance of the topic chosen due to the need to reduce overhead costs and timing of the development of user interfaces, the inability to adapt to the peculiarities of a particular user. As a user of any system communicating with it through an interface, the problems associated with the interface, often form a negative opinion about the system as a whole and does not allow full use of its functionality. This situation is mostly suitable in case of intelligent systems usage as possibilities of such systems is much wider than possibilities of conventional systems.

It is proposed to use ontology-based approach as a basis for the development of intelligent systems user interfaces design technology. The approach involves the development of intelligent systems user interfaces formal ontology. The creation of such ontology promotes coordination of principles and methods for user interface components design at first, promotes unification of user and intelligent system interface activities secondly, promotes decomposition process of user interfaces design and the possibility of its parallelism in third.

B. Problems, Need to be Solved

- the complexity of various kinds of the intelligent systems interfaces leads to the increased time needed to spent on the interface usage training and to learn additional study materials [5];
- long-term design and support of user interfaces and its high costs, what complicates the improvement process and leads to rapid system obsolescence;
- the absence of user interface development process unification what makes it difficult to develop user interfaces in parallel as well as limits the reuse of already developed components;
- long-term user retraining at the stage of learning new intelligent systems interfaces and at the stage of learning new external languages for knowledge representation as a consequence of the absence of such unification;
- there is no ability to use several external languages for knowledge representation simultaneously: various kinds of knowledge may displayed differently if such a display will be convenient for perception. At the same time it should be clear for the user when to use each external language for knowledge representation. In addition there is no possibility of rapid expansion of set of external languages, if it will be necessary;
- difficulties in transferring of user interfaces from one platform to another;
- absence of a common formal basis for constructing models of interfaces void the user's possibility of asking questions related to the interface usage.

C. Analysis of Existing Approaches of Specified Problems Solution

Nowadays there are several scientific researches which aimed to solve the problems of user interfaces development. The user interface is a component of the intelligent system that is prone to frequent changes due to the presence of a wide range of users with different levels of usage culture and requirements for a software system. Create a user interface that satisfies the requirements of all users possible only through a creation of flexible models of such interface, which could then be interpreted by the interpretation of one of the platforms of such models.

In the **classical approach** [14] to create an interface based on user requirements the layout of the interface is built firstly, then the prototype, then, as a rule, determination of the dialogue structure, and work out a possible scenario for the dialogue development, then there is an interface implementation using a suitable programming environment.

Model is the basic of interface development which contains declarative description of a high level of abstraction and does

not contain procedural code in **model-orientied approach** [6]. A set of models is different for each model-oriented tool, for each tool and the level of the model different declarative languages are used, which makes it difficult not only to create an interface in one model-oriented tool, but also its subsequent modification as well as to the strong dependence of the developed model to the tool of its development.

Model-oriented approach based on ontology with specified interface models proposed in [10]. The interface in this approach is focused on the conversion of user-entered information presented in a comprehensible user message, into the values of the application program variables and vice versa. Particular importance is attached to the algorithm of automatic interface model transformation to the code, which is controlled by **user interface ontologies** where the characteristics of a particular model are input to the algorithm. However, this approach does not solve the problem of the user interface complexity and the absence of user possibility to ask questions related to the interface usage.

Another approach is **building an adaptive user interface**. The main focus here is on the user's cognitive characteristics taking into account possibility to create a custom interface that focuses on the modification of the parameters for maximum coordination with a cognitive profile. Adaptation mechanism for the implementation is proposed in [7] to establish a set of software tools including the designer interface, diagnostics subsystem, knowledge base for storing interface settings and individual user information important for building personalized interface.

The next approach for user interfaces design is associated with the **theory of activity** [4]. The user interface is considered as a set of information model of the problem domain, tools and methods of interaction of the user with the information model, and components, ensuring the formation of the information model in the process of a software system in [11]. To reduce a minimum set of user actions usage of **universal command** and intuitive approach to the description of the control elements is proposed in [19]. However, this approach does not consider interfaces extensibility aspect.

By visualizing the knowledge in intelligent systems we mean the mapping knowledge base fragment into the external form. Visualization of knowledge related to the user interface has an important role in knowledge obtaining and transferring. Visual languages complement ontological (or conceptual) modeling technology, making ontology content understandable and intuitive is not only for experts and analysts, but for a beginner [1].

There are several approaches to solving the problem of knowledge visualization. **Visualization of ontologies using cognitive frames** is considered in [16] [17]. The cognitive frame is a visualized fragment of ontology which allows to adequately convey to the person (expert) knowledge related to some target concept. The cognitive frame has two components - the contents of the corresponding ontological context of the target concepts and the visual image, presenting to an expert. The structure of the cognitive frame is based on the use of invariant relationships as well as consideration of concepts at different levels of hierarchy. It allows to form content in the frame that satisfies the requirements of compactness, completeness and familiarity for any ontology concepts.

Various visualization techniques such as hyperbolic trees, conceptual maps, goal-oriented visualisation of relationships between the elements of knowledge are proposed in [8]. Focused on goals integration of contextual visualization of relations between knowledge elements significantly improves the transfer of knowledge from the media (teacher) to recipients (students).

The main disadvantages of existing approaches of visualisation of different types of knowledge are the absence of universal language, which would allow to display any kind of knowledge in th form, though less evident than in the case of a specialized language, but clear to the end user.

As can be noted, the use of the approaches outlined above can solve only some of the identified problems of the user interface design. Several problems such as interface flexibility and effective implementation of new specialized communication languages remain unresolved.

D. The Proposed Approach

Problems of unification of construction of the various components of the computer systems principles are solved within the OSTIS Project [9], aimed at creating an open semantic technology design, managed by knowledge. Systems developed by this technology called **ostis-systems**. In this paper we will talk about ***user interfaces of the ostis-systems***.

As a formal basis for building ostis-system models in this technology is the universal language called ***SC-code***. The texts of the language written in the form of homogeneous semantic networks with set-theoretic interpretation. Elements of such semantic networks called *sc-elements*, such as nodes - *sc-nodes*, connections - *sc-connectors* (*sc-arcs, sc-edges*).

The basis of the ontological user interfaces design are the following principles:

- the user interface is a specialized ostis-system oriented on the interface tasks solution and consisting of the knowledge base and knowledge processing machine for user interface. It allows the user to address the various types of questions to the user interface;
- the ontology-based approach of the user interface design is used. It contributes to defined separation of the activities of various user interface developers as well as the unification of design principles;
- SC-code is used as a formal language of the internal knowledge representation (ontologies, subject domains and others). This ensures ease of interpretation of this knowledge by the system and by the person - the user or developer as well as the perception of the uniqueness of this information perception;
- the syntax and semantics of all kinds of external languages are described using SC-code with the appropriate ontology;
- translation from the internal to the external language and back are organized in such a way that translation mechanisms are not depend on external language. For the implementation of new specialized language in such case specification of the syntax and semantics of the language

is only needed. The universal model of the translation will not be affected by this specification;
- each control element of the user interface is an external display of a sc-element stored into the semantic memory (***sc-memory***). This allows to use them as custom commands arguments and correctly interpret the semantics and pragmatics of the interface objects activity;
- selection of visualisation styles carried out depending on the type of knowledge to display (for example, the use of different visualization elements for certain types of knowledge, and the other - for the other types) is supposed. This allows the user to quickly learn new specialized languages as well as to make a simple and understandable display of knowledge;
- the user interface model built independently of the interpretation of the platform implementation of such a model. It allows an easy transfer of the developed model on different platforms.

The use of the ontology-based approach of user interfaces design involves the construction of (1) the ontological model of the user interface as a specialized ostis-system; (2) ontological model of interfaces design process, those interface developers actions ontology which is based on the proposed model. In this paper the main focus is on the construction of the ontological model of the ostis-system user interface.

Within the described technology knowledge base structure of any ostis-system is described by a hierarchy of *subject domains* and the corresponding *ontologies* [12]. Ontology thus treated as specification of the appropriate domain. So speaking about the development of a certain domain we assume that it is development of an appropriate set of ontologies in the particular.

Ontological model of any entity described by SC-code will call sc-model. Knowledge base model and knowledge processing machine model will be called ***sc-model of knowledge base*** [13] and ***sc-model of knowledge processing machine*** [20] accordingly.

E. Tasks to be Resolved for Proposed Approach Implementation

Clarification of the *intelligent systems user interface ontology-based design* concept proposed in the paper includes solving of the following tasks:

- to develop the *sc-model of the user interface knowledge base* presented by hierarchy of domains and their corresponding ontologies;
- to develop the *sc-model of the user interface knowledge processing machine* presented by family of user interface agents;
- to develop a universal meta-language based on the system of concepts of *Subject domain and ontology of languages and visualization tools* which is the basis for a system of visualization and editing knowledge agents;
- to unify the principles of organisation of user and ostis-system interface activities;
- to unify the intelligent systems user interface commands typology;
- to develop a semantic typology of user interface elements which are the objects of the interface activity;
- to develop a library of reusable components of ostis-systems user interfaces, in which distinguish the *Kernel of the unified models of user interfaces*.

II. User Interface Knowledge Base Structure

As mentioned earlier, the user interface as part of the proposed approach is a specialized ostis-system and therefore knowledge base is its necessary component [12]. The knowledge base of the user interface includes the following parts:

- description of the processes related to the past, present and future of the user interface. Under the previous of the user interface meant the history of its exploitation as well as the evolution of the interface. Under the present - the current state of the user interface. Under the future - user interface development plans. Analysis of described temporal processes allows to evaluate the effectiveness of the interface development and enables versioning of the user interfaces design;
- users models with information about users characteristics, capabilities and preferences, which allows the interface to be flexible and adapt to the user, providing the most efficient interaction;
- typology of user and ostis-systems actions, which allows to describe the principles of interaction of the user interface with users at all levels of the interface interaction;
- typology of the objects of these actions, which allows to produce unification and coordination of user interface components as well as to make their hierarchy;
- formal description of external languages of SC-code structures representation, both universal and specialized.

If we consider the General ontology of user interfaces, it makes sense to speak of a system of four interrelated ontologies:

- ***General ontology of user interfaces***;
- ***General ontology of users and ostis-systems interface actions***;
- ***General ontology of user interfaces developer actions***;
- ***General ontology of users interfaces design tools***.

The first of these describes the common model of ostis-systems user interfaces. The second - a multi-level user interface activities on the one hand and the ostis-system interface activities on the other hand. The third ontology deals with the activities of the developer of ostis-system user interfaces. Finally, the fourth ontology considering various user interfaces design tools, including the various editors, translators and other tools.

This paper will consider the first and second of these ontologies and their corresponding subject domains. The fragments (substructures) of the considered subject domains and ontologies will be further shown in the form of *SC-code text* (***sc-texts***), written in SCn-code [3].

A. Subject domain and ontology of user interfaces

As it was mentioned before, user interface is a specialized ostis-system focused on solving the corresponding class of problems [15]. Further, design of user interface will mean user interface of the ostis-system. Therefore all the principles listed

below will characterize exactly this type of interfaces. Thus, *sc-model* of this interface is built according to general principles of *ostis-systems* construction:

user interface
= ostis-system user interface
= sc-model of ostis-system user interface
<= abstract decomposition:*
{
- *sc-model of ostis-system user interface knowledge base*
- *sc-model of ostis-system user interface knowledge processing machine*

}

A concept of the *ostis-system user interface*, different user interface elements like windows, controls, states which describe user interface are investigated in the context of the *Subject domain of user interfaces*.

The key concepts of this domain model are:

- concepts, which denote different *classes of the user interfaces*;
- concepts, which denote different objects on which the user interface activity may be directed;
- relations defined on the set of interface activity objects.

Maximum studied object class of analysed subject domain and ontology is the concept of the *ostis-system user interface*.

The following subclasses of user interfaces are defined based on the type of interaction [18]:

user interface
=> inclusion:*
- *command user interface of IMS*
  = interface where commands are given to a computer system using the command line and then transferred for execution
- *graphical user interface*
  = interface where commands are given to a computer system indirectly through graphic images
- *SILK-interface*
  = interface where commands are given to a computer system through the analysis of human behavior
  => inclusion:*
  - *natural language interface*
    => inclusion:*
    - *speech interface*

Each intelligent system operates with knowledge base using internal language. A dialog is represented as a message exchanging between user and intelligent system. So as such a dialog takes place, a fragment of knowledge base should be mapped into some external form.

The universal external language of message exchanging is a language which allows to describe any kind of knowledge. The SC-code and all its representations are examples of this language:

- **SCg-code** - is one of possible languages to visualize SC-texts. The basic principle of SCg-code is that each sc-element is mapped into scg-element(a graphic representation);
- **SCs-code** – a string (linear) representation of SC-texts. It visualizes sc-text as a sequence of characters;
- **SCn-code** – a string nonlinear representation of SC-texts. SCn-code visualizes sc-text in a formatted with special rules sequence of characters in which also can be used basic tools of hypermedia, images and tools for navigation between parts of scn-texts.

The specialized language of an external messaging is a language which intended to describe particular forms of knowledge applicable in specific areas of science and technology. The most widespread examples of this language are language of drawings and language of description of topographic maps.

Geometric primitives are the main elements for language of drawings. This language is used in the geometry intelligent system which is built with help of ostis-technology.

Terrain elements and their properties are the main elements for language of description of topographic maps. This language can be reused in various ostis-systems.

Message exchange is performed with help of user interface actions in the ostis-system. Also it's possible to interact with some objects on the screen. Controls and windows are examples of such objects.

Each ***control*** represents a class of actions which can be initiated by user. Using such actions user has an influence on visible objects and on state of sc-elements associated with these objects.

SC-elements can be divided into groups and marked during visualization(e.g. with some color). The possible groups of sc-elements are:

- sc-elements which have been just generated on the screen by user;
- sc-elements which have been just generated by user and are available in the knowledge base;
- sc-elements which have been just generated by user and are not available in the knowledge base;
- sc-elements which have been displayed on the screen by ostis-system from the knowledge base.

Localization of interaction between a user and ostis-system can be performed via user interface ***windows***. The types of possible ostis-system windows are shown below:

window
= ostis-system window
= sc-text sign or file sign (which may be empty) stored in sc-memory at the current time and displayed on the screen
=> inclusion:*
- *main window of ostis-system*

<= partitioning:*
{
- *frame window of ostis-system*
- *contour window of ostis-system*

}

contour window of ostis-system
= (scg-contour ∩ ostis-system window)

= *scg-contour which is the ostis-system window displayed on the screen and can be manipulated*

frame window of ostis-system
= *(scg-frame ∩ ostis-system window)*
= *scg-frame which is the ostis-system window displayed on the screen and can be manipulated*

The ***main window*** is distinguished among *all windows in the ostis-system* based on the principle of inheritance of windows. This window is not a child of any window displayed on the screen.

It has the following properties:

- Only one instance of main window is available for a user session in the ostis-system;
- Main window belongs to the contoured set of elements;
- It's not possible to remove or hide main window.

Relation 'child window'* is an example of concept which is investigated in scope of *Subject domain of user interfaces*. This relation helps to build hierarchy of *windows in the ostis-system*.

Each tuple of the relation consists of window sc-elements and has the following properties:

- Oi child window must be a part of the (be in) parent window Oj;
- There is no such a window Ok, which is a parent for Oi and Oj windows at the same time.

User interface of ostis-system goes from one state into another, defines set and appearance of objects which are visible on the screen. Variety of these states defines a mode of user interface. It makes user interface flexible and helps to distinguish actions which are available for different categories of users.

mode
= *situational set of states of the user interface, which state is considered during executinon of user action.*
⊂ *situational set*
=> *inclusion*:*
- *mode of identification language*
- *mode of language for visualization of external texts*
- *mode for displaying sequences of messages*

The description of each mode is listed below:

- ***mode of identification language*** - the *mode* which set a specific *language for the identification* of displayed object.
  Russian and English identification languages are supported in current version of technology.
- ***mode of language for visualization of external texts*** - the *mode* which set a specific *external language to display selected texts*.
  SCn-code, SCg-code, language of drawings and language of description of topographic maps are supported in current version of technology.
- ***mode for displaying sequences of messages*** - the *mode* which set a specific *visualization sequences of messages** during user interaction with system.
  Display all messages in single window and display each message in new window are supported modes in current version of technology.

B. Subject domain and ontology of users and ostis-systems interface actions

Subject domain and ontology of users and ostis-systems interface actions
<= *sc-structure decomposition*:*
{
- *Subject domain and ontology of ostis-systems users interface actions*
- *Subject domain and ontology of ostis-systems interface actions*

}

1) Subject domain and ontology of ostis-systems users interface actions: Interface language of ostis-system users, like any other language, has its own syntax and semantics and represents a set of a certain kind of texts. The text of such language is a sequence of interface actions of ostis-system users.

The proposal (the minimum meaningful piece of text) of interface language of ostis-system users is the specification of some initiated ostis-system action or the command formulation addressed to ostis-system. In this case, if an action initiated by user [21] does not require the indication of action objects (arguments), the proposal will consist of one elementary interface action of ostis-system user, which is an indication of the type of initiated action.

Development of ***Subject domain and ontology of ostis-systems users interface actions*** solves the problem of a clear separation of the activities of the user interfaces developers for effective and rapid development and facilitates the unification of user interface design principles.

The objects of research in *Subject domain and ontology of ostis-systems users interface actions* in the context of the general typology of actions are elementary user actions, interface commands and ostis-system users messages.

The key concepts of this domain are:

- concepts, indicating different *classes of elementary user actions*, interface commands and *ostis-system users messages*;
- relationships defined on the set of elementary user actions and interface command, related to the *Subject domain and ontology of actions and tasks*;
- relationships defined on the set of ostis-system users messages, related to the *Subject domain and ontology of temporal entities*;
- relationships defined on the set of ostis-system users messages, which can be defined only in scope of *Subject domain and ontology of ostis-systems users interface actions*.

The maximum class of research objects of considered subject domain and ontology is the concept of *ostis-system user interface action*.

On the basis of atomacity (elementary quality) class of shown actions is divided into the next subclasses:

interface action of an ostis-system user
<= union:*
{
- *elementary interface action of an ostis-system user*
  = interface action of an ostis-system user, for which another ingoing in his composition interface actions of the ostis-system user don't exist
- *interface command of ostis-systerm user forming*
- *message of ostis-systerm user forming*

}

The simpliest interface language fragments of ostis-system users are *elementary interface actions of ostis-system users*, for which another ingoing in his composition interface actions of the ostis-system user don't exist. Notice that the description of each elementary user action is abstract, so independence of the actions from implementation on various devices is provided [2].

Like atomic text fragments of any language, elementary action of ostis-system user is associated with its own alphabet (syntactically recognizable typology) below:

elementary interface action of ostis-system user
= the interface action of ostis-system user, for which another ingoing in his composition interface actions of the ostis-system user don't exist
⊃ specification of the type initiated by ostis-system user
⊃ specification of the argument (object) initiated by ostis-system user
⊃ specification of the entity, for which base decomposition is requested
⊃ specification of the of completion of arguments transfer
⊃ cancel of the last elementary action of ostis-system user
⊃ cancel of all specified (listed) arguments

Among the listed subclasses of elementary actions of an ostis-system user special attention should be paid to the action of *specification for the entity, for which the base decomposition is requested*, because the action is an interface user command at the same time. Because output process of the decomposition (base semantic neighborhood) for some entity is the most frequent user request, the action included to the list of elementary user actions, that makes a task of explicit using of the base decomposition request command with specification for its arguments easier.

At the same time, it is worth remembering that not every elementary user action is an interface command, which will be discussed further.

an interface command of ostis-system user
= specification for ostis-system user action, formed in the language of interface actions of ostis-system users and included type specification for an initiated action and specification for its arguments, i.e. objects, for which the action must be executed

The next kinds of forming of an interface command allowed depending on whether the user addresses to the signs of initiated actions by them or uses resources of external editors, that allows to reproduce specifications of initiated actions:

interface command of ostis-system user
<= partitioning:*
{
- *command, formed on interface actions language*
  = command, commiting the fact, that specification for an executing action generates automatically by applying of elementary user actions to the control elements
- *command, formed on external language*
  = command, commiting the fact, that specification for an executing action generates whether by the user, using resources of special editor, or using instraction on natural language

}

Interface actions of ostis-system users are subclasses of regular actions, therefore according to the ontology approach, in particular, the existing system of ontologis and subject domains are possible crossing of subject domains, the object of study of which are this two mentioned concepts. Examples of the relations from *Subject domain and ontology of actions and tasks* in the considered domain are such relations as ***object****, ***result****, ***context of action****, etc.

The most difficult, in terms of structure, interface actions class of ostis-system users is message forming.

message of ostis-system user
<= partitioning:*
{
- *user message in the external language*
  = message generated in the language of the interface actions (interface commands), which represents a sequence of actions with the indication of the objects on which these actions are defined, and the types of actions
  => inclusion:*
    - *scn-message*
    - *scg-message*
- *user message on the internal language*
  = sc-message
  = sc-text, representing the meaning of information built by one subject and intended for use by some particular subject or a group of subjects

}

Depending on what kind of sense carries the generated message, there are following subclasses of message of ostis-system user:

message of ostis-system user
<= partitioning:*
{
- *narrative message*
  = message provided the recipient some information and does not require any additional user action.*
- *imperative message*
  = message, which suggests some action by the recipient after receiving of the message.*

Some interface commands can simultaneously be imperative messages of the ostis-system users, in this case is said about message, which is decorated in the language of the interface commands. However, the main difference between them is that the message directly affects to the sc-elements, i.e, the formation of ostis-system user message means entering this message in the substantive part of the knowledge base.

Moreover, messages movement from one form to another also depends on the used commands. Commands-requests are initiated at once, in this case the message is converted from one form to another almost immediately, commands of editing form the actions protocol connected with the creation of the message content in one of the foreign languages, then the message is translated to the ostis-system memory that runs the process of initiating sequential actions from a protocol and the subsequent integration of the content of the message with the substantive part of the knowledge base.

Specifications of actions, which are the interface commands, does not always mean the use of sc-elements as arguments. For example, when it comes to the commands of editing file, the use of these commands affects the content of these files, but does not affect the sign of the file. However, if you imagine the content of a file as a linked sc-text, the reference to a certain subset of the sc-text (e.g, the sign of the proposal, the sign of some letter) will already imply the formation of user message.

Relations specified on messages consider the temporary connections between them, for example, *sequence of messages** for a particular user during the session of ostis-system operation. The existence of such relations indicates the intersection of the analysed subject domain and *Subject domain of temporary entities*.

Consider the examples of such relations:

sequence of messages*
=> *inclusion**:
- *temporary sequence of messages**
- *logical sequence of messages**
  => *inclusion**:
  - *answer**
    = *binary oriented relation, the first component of the ligament of which is a sign of formulated in the form of an imperative message sender* request, and the second component - a sign of narrative message in response to this request.*
  - *message that specifies statement of the problem**
    = *binary oriented relation, the first component of the ligament of which is a sign of formulated in the form of an imperative message sender* request, and the second component — a sign of recipient* imperative message if the sc-text of the original message is not completed.*
  - *error message**
    = *binary oriented relation, the first component of the ligament of which is a sign of formulated in the form of an imperative message sender* request, and the second component - a sign of narrative message, representing a formed incorrect structure by the result of request processing.*

Completing the description of the interface actions of ostis-system users, consider some of the relations specific only for *Subject domain and ontology of ostis-systems users interface actions* that reveal deeper aspects of visualization and interaction with the outside world:

- ***outer form of message**** - *binary oriented relation*, the first component of the ligament of which is the *sign of message*, and the second component - a *sign of the file* generated as a result of *sc-text translation** to the external language;
- ***communication environment**** - *binary oriented relation*, the first component of the ligament of which is the sign of message, and the second component - a sign of a certain *subset* of the surrounding environment, in which the messages exchanged between the *sender** and the *recipient**. As a *communication environment** for *ostis-systems* is *sc-memory*.

2) Subject domain and ontology of ostis-systems interface actions: Frequently, in spite of the same appointment of software and the similarity of tasks solved by them, from the user point of view interfaces of such tools externally looks very different, leading to the need for user to retraining and adapting to the new principles of interaction with the system. The solution of the problem is to explicitly allocate semantic types and classes of actions in a variety of ostis-systems interface actions and reflect by the semantics of the syntactic (visual) separation signs of such actions when they are displayed on the screen.

Development of ***Subject domain and ontology of ostis-systems interface actions*** solves the problem of correct and unambiguous interpretation of semantics and pragmatics of elements displayed on the screen This allows to obtain comprehensive information about the purpose and use of objects of interface activity at lower levels of working with it.

The objects of research in Subject domain and ontology of ostis-systems interface actions in the context of the general typology of actions is ostis-system direct action initiated by users and indirect actions the implementation of which is implicit in the execution of other actions.

The key concepts of this domain are:

- concepts, indicating different *classes and types of ostis-systems interface actions*;
- relationships defined on the set of ostis-system interface actions related to the *Subject domain and the ontology of actions and tasks*.

The maximum class of objects of research of the considered subject domain and ontology is the concept of *ostis-system interface action*.

ostis-system interface action
<= *partitioning**:
{
- *atomic ostis-system interface action*
  = *ostis-system interface action performed by atomic agent*
- *non-atomic ostis-system interface action*

= *ostis-system interface action which is a sequence of atomic ostis-system interface actions and performed by non-atomic agent*
}

At the stage of interaction with the elements on the screen ***non-atomic ostis-system interface action classes*** are allocated, the interpretation of which is to provide a list of ***types of user-initiated actions*** relating to this class. Syntax highlighting for classes and types of action is supposed to intuitively understand the purpose of the user interface control elements implies a signs of sc-elements mentioned above.

Speaking about the classification of ostis-system interface actions in terms of their purpose, in this case it is preferable to use a color selection or the introduction of a specified extension as an option for the external representation (for example, SCg-code), which includes into the alphabet primitives associated with different classes of operations.

class of ostis-system actions, initiated by its users
<= *partitioning*:*
{
- *non-atomic class of ostis-system actions, initiated by its users*
- *type of ostis-system actions, initiated by its users*

}
<= *partitioning*:*
{
- *action of interpreting the program stored into sc-memory*
- *information searching action*
  => *inclusion*:*
  - *information searching action into the whole knowledge base*
  - *information searching action into the specified knowledge base fragment*
- *action of knowledge base editing*
- *action of editing file stored in sc-memory*
- *action of ostis-system building*
- *action of ostis-system operation mode setting*
- *windows manipulation action*
- *ostis-system action in the external environment*
- *action of cancelling the latest initiated ostis-system action*

}

In case when the number of initiated action arguments is important, the introduction of syntax rules here is not even a recommendation, but a necessity as receiving information on the number and type of action arguments can not be obtained without recourse to the appropriate action specification. As in the case with the appointment of interface actions syntax highlighting the best option here is to use some variant of the external representation. A complete classification of the types of actions initiated by ostis-system user will be given further:

type of actions initiated by ostis-system user
= *elementary class of ostis-system actions*
= *class of ostis-system actions an indication of which together with the corresponding action arguments (object on which the action is performed) uniquely defines every ostis-system action*
<= *partitioning*:*
{
- *type of actions initiated by ostis-system user with fixed arguments count*
  <= *partitioning*:*
  {
  - *type of actions initiated by ostis-system user without arguments*
  - *type of actions initiated by ostis-system user with one argument*
  - *type of actions initiated by ostis-system user with two arguments*
  - *type of actions initiated by ostis-system user with three and more arguments*

  }
- *type of actions initiated by ostis-system user with any arguments count*

In addition to ostis-system user messages messages of ostis-system itself are distinguished:

ostis-system message
=> *inclusion*:*
- *ostis-system message as a response on the imperative ostis-system user message*
- *ostis-system message initiated by itself*

Possible ostis-systems reactions to the imperative message are:

- an indication of the fact of the completion of certain task. Is typical for behavioral actions for example;
- receiving a response on the assigned task generated either as a result of the of the user interface knowledge base analysis or as a result of analysis of the subject part of the ostis-system knowledge base itself.

Situations in which ostis-system initiates communication itself can be divided into two groups:

- situations arised in the analysis of user activity.
  Examples: assignment arguments, which are not correspond the type of initiated action; hints appearance during the usage of user interface elements.
- situations arised in the analysis of the syntax of the external language texts.
  Examples:incompleteness of sentences formed on the external language; usage of structures, atypical or incorrectly used in the context separately taken external language.

Based on the description of the main entities that reflect the ostis-system interface activities, will describe the scenario of messaging between the user and ostis-system on the example of the work with file.

- receptor agents which fix the fact of performing elementary user actions on a file and generates a sequence of actions in time;
- effector agent recognizes in the generated sequence interface user commands and defines order of execution of these commands;

- the input of formed user message is happening on the external language;
- effector agent of translation the file to SC-code converts (if possible) the contents of the file as a sequence of sc-elements constituting the fragment of connected sc-text unambiguously interpreted in ostis-system memory;
- The formation of ostis-system response message on the external language is happening which was formed by a user message and taking into account the mode of the user interface which is currently set for the user interface.

The same relationships can be set on the ostis-system user interface actions as in the *Subject domain and ontology of actions and tasks* that have been mentioned above in the *Subject domain of user interface actions*. In addition, such taxonomic relations, showing a more complex organization of ostis-systems interface such as ***sub-action****, ***action decomposition**** and others can be used.

A special case of equivalence relation is also introduced for interface actions:

equivalence of the ostis-system actions, initiated by its users*
∈ *equivalence relation*
= *relation between the actions which have different specifications but the same result*

III. Structure of user interface knowledge processing machine

User interface knowledge processing machine consists of some collective of sc-agents [12] that provide user experience with control element of ostis-systems. Consider a model of user interface knowledge processing machine as an aggregate of abstract sc-agents that are meant by some software implementation.

abstract sc-agent of user interface
<= *abstract sc-agent decomposition**:
{
- *Abstract sc-agent of external texts editing*
  => *inclusion**:
  - *Abstract sc-agent of ostis-system's window manipulating*
    <= *abstract sc-agent decomposition**:
    {
    - *Abstract sc-agent of new window creating*
    - *Abstract sc-agent of window closing (minimizing)*
    - *Abstract sc-agent of window removing*
    - *Abstract sc-agent of window moving*

    }
- *Abstract sc-agent of mode setting*
  <= *abstract sc-agent decomposition**:
  {
  - *Abstract sc-agent of identification language setting*
  - *Abstract sc-agent of external texts displaying language setting*
  - *Abstract sc-agent of message sequence displaying*

  }
- *Abstract sc-agent of work with commands*
  <= *abstract sc-agent decomposition**:
  {
  - *Abstract sc-agent of displaying semantic neighborhood of command*
  - *Abstract sc-agent of search of commands, which are members of one class*
  - *Abstract sc-agent of search of command class that matches specified command*
  - *Abstract sc-agent of search of commands, for which given entity can be an argument*
  - *Abstract sc-agent of command recognition from the set of elementary interface actions*
  - *Abstract sc-agent of last executed user command cancellation*

  }
- *Abstract sc-agent of file translation into external language*
  <= *abstract sc-agent decomposition**:
  {
  - *Abstract sc-agent of file translation into SCn-code*
  - *Abstract sc-agent of file translation into SCg-code*
  - *Abstract sc-agent of file translation into specialized language*

  }
- *Abstract sc-agent of fixation of elementary user action execution fact*
- *Abstract sc-agent of sc-message translation into external language*
- *Abstract sc-agent of last executed user command cancellation*

}

If you use agents, It will worth to remember the differences in semantic and pragmatic component of any element of an user interface. Semantic component consists of determining sign of witch entity is the element displayed on the screen. Pragmatic component considers applied aspect (aspect of application) of the element displayed on the screen.

Only semantic component matters on sc-memory level. However this fact does not influence on exploitation process of system by user, because both components reflect different sides of the same sign of some entity. For example, every localized sc-text of some *message* is hidden behind *window of ostis-system*, every button hides behind itself sign of some *actions class* initiated by pressing input device keys.

Initiation event for two groups of abstract sc-agents - *Abstract sc-agents of ostis-system's window manipulating* and *Abstract sc-agents of mode setting* - will occur in three cases:

- in connection with keystroke or after selection of specific item in a drop-down list (in this case input parameters of sc-agent are taken by default);
- in connection with using command with previously specified arguments (in this case user choose specific control element as sc-agent's input parameter);
- in connection with generation of action specification and inclusion of this action into set of initiated actions (in this

case user knows set-theoretic interpretation of abstract sc-agent).

The third group of abstract sc-agents - *Abstract sc-agent of work with commands* - is a particular case of search sc-agents, whose task is to view the full or partial semantic neighborhood of the key element of this neighborhood or to find connections between the key element and other entities in the ostis-system knowledge base. These agents don't imply default arguments. It is the reason why only second and third of outlined above ways of initiating are valid for them.

The fourth group of abstract sc-agents - *Abstract sc-agents of file translation into external language* and connected to them *Abstract sc-agent of sc-message translation into external language* and *Abstract sc-agent of sc-message translation into external language* - is involved in messaging between user and ostis-system. These agents are not added into a list of commands that are available to a user as a menu item or as an user interface element because they are initiated by ostis-system. Therefore outlined above initiating events are not applied to these agents.

Finally *Abstract sc-agent of last executed user command cancellation* allows to correct activity of ostis-system's user by retrieval through protocol of executed action in case of committing unintentional errors by user.

IV. Library of reusable components of user interfaces

User interface component is an important concept of this paper.

user interface component
= a limited set of user interface elements that is unified and that can be used repeatedly in various intelligent systems

Each *component of the user interface* conforms to a fragment of the knowledge base based on some set of concepts discussed earlier and a set of *sc-agents*, manipulating this fragment in *sc-memory*. Depending on tools one component was developed with, the following subclasses are marked:

user interface component
*<= partitioning**
{
- *platform-independent user interface component = user interface component developed and maintained on OSTIS Technology tools.*
- *platform-dependent user interface component = user interface component developed with using of extraneous tools that are different from OSTIS Technology tools*

}

Platform dependent components of the user interface get through a process of integration to function properly as a part of ostis-system. In the course of this process the debugging of this component and sc-memory interaction is made. In the simplest situation it's provided by building of translators to sc-memory and from it.

One of the interfaces designing complication is a platform orientation of interfaces. To solve this problem, the components that realize ***the user interface kernel of intelligent system*** are included in the library of reusable components of a user interface. It consists of knowledge base model containing some subset of the knowledge base of a user interface and a model of a minimum essential knowledge processing machine.

V. Conclusion

The paper reviews the ontological approach to user interface design which is based on presentation of the user interface as specialized embedded intelligent subsystem that is intended for realization of information exchange between intelligent system and its users.

The use of this approach offers the following advantages:

- flexibility of designed interfaces, its maintenance and enhancement.
- time development lowering due to:
  - possibility of the interface activity separation in the design of user interfaces and minimizing the number of agreements in the process of collective development;
  - accumulation and use of project experience (design solutions) of other developers that is presented in the form of the user interface specified components as part of component libraries;
  - the use of unified approach to design of both the user interface and the interface of the ostis-system, its users and developers activity.
- portability of resultant ontological model of the user interfaces for different platforms
- improvement of information perception, that is displayed on the screen, through the use of general-purpose and specialized languages of external texts representation described by a common formal basis - SC-code.

Intelligence of the user interface is expressed in:

- correctness and efficiency analysis of user actions;
- serving out recommendations to user in case of incorrect and inefficient actions;
- identification of user commands, which can lead to dangerous (irreversible) harmful effects: in this case the request of additional action confirmation is carried out;
- formation of response to any user question that concerns the interface activity organization.

Thanks to presentation of the user interface in the form of ostis-system, it is possible to address different queries to the interface, it means, to use it as an all-round help-system, as well as to use objects (sc-elements), that are displayed on the screen, as arguments to any queries that may be useful, for example, when you configure the user interface for user specific needs.

In this way, the screen of the user interface displays the knowledge base fragment that is stored in the memory of ostis-system, though the user interface control elements are appropriate sc-elemets visualization, which adds the conciseness to the interface.

This work was supported by BRFFR-RFFR (Ф15РМ-073).

Список литературы

[1] Gavrilova, T.A. Evaluation of the cognitive ergonomics of ontologies on the basis of graph analysis / T.A. Gavrilova, V.A. Gorovoy, E.S. Bolotnikova // Scientific and Technical Information Processing, December 2010, Volume 37, Issue 6. - P.398-406.

[2] (2016, Nov.) Extensible User Interface Language. [Online]. Available: xml.coverpages.org/xul.html

[3] (2016, Nov.) The IMS.OSTIS website. [Online]. Available: http://www.ims.ostis.net/

[4] Mikko Korpela, Anja Mursu, H.A. Soriyan. Information Systems Development as an Activity // Computer Supported Cooperative Work 11. - 2002. - P. 111-128.

[5] Patrick, 2003. Intelligent user interfaces: introduction and survey / Patrick A.M. – Delft University of Technology, 2003.

[6] Szekely P., Sukaviriya P., Castells P., Muthukumarasamy J., Salcher E. Declarative Interface Models for User Interface Construction Tools: the Mastermind Approach. In Engineering for Human-Computer Interaction, L. Bass and C. Unger Eds. Chapman & Hall, 1996.

[7] Белоусова С.А., Рогозов Ю.И. Анализ подходов к созданию пользовательского интерфейса. «Известия ЮФУ» №6. 2014. – С.142-148.

[8] Бодров В.Н., Магалашвили В.В. Ориентированная на цели визуализация знаний // Международный журнал «образовательные технологии и общество». – 2008. – Т. 11, № 1. – С. 420-433.

[9] Голенков В.В., Гулякина Н.А. Семантическая технология компонентного проектирования систем, управляемых знаниями // Открытые семантические технологии проектирования интеллектуальных систем: материалы V Междунар. Науч.-тех. конф./редкол.: В.В. Голенков [и др.]. – Минск: БГУИР, 2015.

[10] Грибова В.В., Клещев А.С. Концепция разработки пользовательского интерфейса на основе онтологий. Ч. 1. Инструментарий для разработки пользовательского интерфейса (обзор литературы). Основная идея подхода. - Владивосток: ИАПУ ДВО РАН, 2003. -24 с.

[11] Гультяев А.К., Машин В.А. Проектирование и дизайн пользовательского интерфейса.- СПб.: КОРОНА принт, 2007. - 239 с.

[12] Давыденко И.Т., Гракова Н.В., Сергиенко Е.С., Федотова А.В. Средства структуризации семантических моделей баз знаний // Открытые семантические технологии проектирования интеллектуальных систем: материалы VI Междунар. Науч.-тех. конф./редкол.: В.В. Голенков [и др.]. – Минск: БГУИР, 2016.

[13] Давыденко, И.Т. Семантическая модель коллективного проектирования баз знаний / И.Т. Давыденко // Открытые семантические технологии проектирования интеллектуальных систем (OSTIS-2016): материалы VI Междунар.научн.-техн.конф. – Мн.: БГУИР, 2016.

[14] (Nov, 2016) Копылов А. Чего не хватает Microsoft Blend: взгляд проектировщика взаимодействия. [Online]. – Available: http://www.gui.ru/copylove/xaml-for-interction-design/.

[15] Корончик Д.Н. Семантическая технология компонентного проектирования пользовательских интерфейсов интеллектуальных систем // Открытые семантические технологии проектирования интеллектуальных систем: материалы I Междунар. Науч.-тех. конф./редкол.: В.В. Голенков [и др.]. – Минск: БГУИР, 2011.

[16] Ломов П. А., Шишаев М. Г. Визуализация OWL-онтологий на основе когнитивных фреймов. - «Труды Кольского научного центра РАН» №5 (18). 2013 - С. 77-89.

[17] Ломов П.А., Шишаев М.Г., Данилов Е.Ю. Визуализация на основе когнитивных фреймов для передачи знаний. - Минск: Изд. центр БГУ. 2015 - С. 35.

[18] Мелешко, А. Д. Интерфейсы взаимодействия человека и компьютера / А. Д. Мелешко // Современные компьютерные информационные технологии: тезисы XI Межвузовской научной студенческой конференции, [Минск], 21 апреля 2010 г. - Минск: БГЭУ, 2010. - С. 117-119.

[19] Раскин Д. Интерфейс: новые направления в проектировании пользовательских интерфейсов. - Символ-Плюс, 2005 - 272 с.

[20] Шункевич Д.В. Машина обработки знаний интеллектуальной метасистемы поддержки проектирования интеллектуальных систем // Открытые семантические технологии проектирования интеллектуальных систем: материалы IV Междунар. Науч.-тех. конф./редкол.: В.В. Голенков [и др.]. – Минск: БГУИР, 2014.

[21] Шункевич, Д.В. Формальное семантическое описание целенаправленной деятельности различного вида субъектов / Д.В. Шункевич, А.В. Губаревич, М.Н. Святкина, О.Л. Моросин // Открытые семантические технологии проектирования интеллектуальных систем (OSTIS-2016): материалы VI Междунар.научн.-техн.конф. – Мн.: БГУИР, 2016.

ОНТОЛОГИЧЕСКОЕ ПРОЕКТИРОВАНИЕ ПОЛЬЗОВАТЕЛЬСКИХ ИНТЕРФЕЙСОВ ИНТЕЛЛЕКТУАЛЬНЫХ СИСТЕМ

Борискин А.С., Корончик Д.Н., Жуков И.И., Садовский М.Е., Хусаинов А.Ф.

В работе рассмотрен онтологический подход к проектированию пользовательских интерфейсов, в основе которого лежит представление пользовательского интерфейса в виде специализированной встроенной интеллектуальной подсистемы, предназначенной для реализации обмена информацией между интеллектуальной системой и её пользователями.

Для достижения поставленной цели необходимо решить следующие проблемы:

- сложность интерфейса интеллектуальных систем различного рода приводит к затратам времени на обучение использованию таких интерфейсов и изучение дополнительных материалов;
- велики сроки разработки и затраты на проектирование и поддержку пользовательских интерфейсов, что осложняет процесс их совершенствования и приводит к их быстрому моральному старению;
- отсутствие унификации в принципах построения пользовательских интерфейсов затрудняет возможность распараллеливания процесса проектирования пользовательских интерфейсов, а также ограничивает возможность повторного использования уже разработанных компонентов;
- как следствие отсутствия такой унификации, велики сроки переобучения пользователя на этапе освоения новых интерфейсов интеллектуальных систем и на этапе освоения новых внешних языков представления знаний.
- отсутствует возможность одновременного использования нескольких внешних языков внешнего отображения хранимых в системе знаний: различные виды знаний могут отображаться по-разному, если такое отображение будет более удобным для восприятия. Кроме того отсутствует возможность быстрого расширения набора внешних языков при необходимости;
- затруднена возможность переноса пользовательских интерфейсов с одной платформы реализации

на другую;
- отсутствие общей формальной основы при построении моделей интерфейсов лишает пользователя возможности задания вопросов, касающихся организации самого интерфейса.

Проблемы унификации принципов построения различных компонентов компьютерных систем решаются в рамках Проекта OSTIS, направленного на создание открытой семантической технологии проектирования систем, управляемых знаниями. Системы, разрабатываемые по данной технологии, названы ostis-системами.

В основе онтологического проектирования пользовательских интерфейсов лежат следующие принципы:

- пользовательский интерфейс представляет собой специализированную ostis-систему, ориентированную на решение интерфейсных задач;
- используется онтологический подход к проектированию пользовательского интерфейса, что способствует чёткому разделению деятельности различных разработчиков пользовательских интерфейсов, а также унификации принципов проектирования;
- используется SC-код в качестве формального языка внутреннего представления знаний (онтологий, предметных областей и др.), благодаря чему обеспечивается легкость интерпретации этих знаний и системой, и человеком - пользователем или разработчиком;
- средствами SC-кода с помощью соответствующих онтологий описываются синтаксис и семантика всевозможных используемых внешних языков;
- трансляции с внутреннего языка на внешний и обратно организовываются так, чтобы механизмы трансляции не зависели от внешнего языка;
- каждый элемент управления пользовательского интерфейса является внешним отображением некоторого элемента, хранящегося в семантической памяти;
- предполагается выбор стилей визуализации, осуществляемый в зависимости от вида отображаемых знаний;
- модель пользовательского интерфейса строится независимо от реализации платформы интерпретации такой модели.

Использование онтологического подхода к проектированию пользовательских интерфейсов предполагает построение (1) онтологической модели самого пользовательского интерфейса, как специализированной ostis-системы; (2) онтологической модели процесса проектирования интерфейсов, т.е. онтологии действий разработчиков интерфейсов, построенных на основе предлагаемой модели. В рамках данной работы внимание уделено построению онтологической модели пользовательского интерфейса ostis-системы.

В рамках описываемой технологии структура базы знаний любой ostis-системы описывается иерархией предметных областей и соответствующих им онтологий. Онтология при этом трактуется как того или иного рода спецификация соответствующей предметной области. Таким образом, при разработке некоторой предметной области речь идет, в том числе, о разработке соответствующего набора онтологий.

Использование данного подхода даёт следующие преимущества:

- гибкость проектируемых интерфейсов, простота их поддержки и совершенствования;
- снижение сроков разработки пользовательских интерфейсов за счёт:
  - возможности разделения интерфейсной деятельности при проектировании пользовательских интерфейсов и минимизации числа согласований в процессе коллективной разработки;
  - накопления и использования проектного опыта (проектных решений) других разработчиков, содержащегося в виде специфицированных компонентов пользовательских интерфейсов в составе библиотеки таких компонентов;
  - использования унифицированного подхода к проектированию как самих пользовательских интерфейсов, так и к интерфейсной деятельности ostis-системы, её пользователей и разработчиков.
- переносимость сформированной онтологической модели пользовательских интерфейсов на различные платформы
- улучшения восприятия информации, отображаемой на экране благодаря использованию универсальных и специализированных языков представления внешних текстов, описанных с помощью единой формальной основы - SC-кода.

Интеллектуальность пользовательского интерфейса выражается в следующем:

- анализ корректности и эффективности пользовательских действий;
- выдача пользователю рекомендаций в случае его некорректных и неэффективных действий;
- выявление пользовательских команд, которые могут вызвать опасные (необратимые) или вредные последствия: в этом случае осуществляется запрос дополнительного подтверждения их выполнения;
- формирование ответа на любой вопрос пользователя, касающийся организации интерфейсной деятельности.

Дополнительными преимуществами представления пользовательского интерфейса в виде ostis-системы является возможность адресовать интерфейсу вопросы различного рода, то есть, использовать его как полноценную help-систему, а также использовать объекты отображённые на экране, в качестве аргументов любых запросов, что может быть полезным, например, при настройке пользовательского интерфейса под нужды конкретного пользователя.