Ontological approach to the building of semantic models of user interfaces

Sadouski M.E.
Belarusian State University of Informatics and Radioelectronics
Minsk, Belarus
sadovski@bsuir.by

***Abstract*—The article is dedicated to the description of the ontological approach to the building of the user interface based on the OSTIS Technology. The existing approaches in the field of the building of user interfaces are considered and an ontological model of support of the building process is described.**

***Keywords*—OSTIS, user interface, building of the user interface, ontological approach, semantic model, server-driven user interface**

I. Introduction

In the modern world and the daily life of humans, there is a growing need for the usage of computer systems. The effectiveness of their usage depends largely on the user interface, since it is the user interface as a component of the system that is a way to interact with the user.

At present, the user interface is the most frequently changed component of the system and is the part of the application that requires the maximum number of updates. Approximately 80% of the costs in the development of computer systems are accounted for by the design, testing and development of the user interface. At the same time, almost all applications change their interface after the release of the first version and the addition of new functionality always affects the already developed component. In addition, most modern systems should be cross-platform for the convenience of users, which implies the development of a web version of the interface as well as mobile and desktop versions [1].

When building the user interface, the following problems remain relevant:

- the portability of user interfaces from one implementation platform to another is difficult;
- the lack of general principles for building user interfaces limits the possibility of reuse of already developed components and increases the time required to train the user in new user interfaces, which also increases the development time and the cost of designing and supporting user interfaces;
- in most systems, there is no possibility of modifying the user interface during running;
- in most systems, there is no possibility of flexible adaptation of the user interface to the needs of a particular user.

Within the framework of this article, an ontological approach to the building of semantic models of user interfaces based on the OSTIS Technology is proposed to solve the above problems. A technique for developing a user interface is also proposed. The article elaborates the ideas proposed in [2] and is aimed at more detailed consideration of the problem of the automatic building of the user interface, which is a key one to support the flexibility and simplicity of improving the designed interfaces.

The building of the user interface within the framework of the proposed approach will be carried out on the basis of its complete semantic model that contains a precise specification of all the used concepts with the help of a hierarchical system of formal ontologies, which will ensure the integration of various aspects of the user interface within a unified system, the ability of the system to analyze the actions performed within the user interface and its flexible configuration in the process of operation. Thus, the development of the user interface will be reduced to the building and improvement of its semantic model.

II. Analysis of existing approaches

Currently, there are several basic approaches to generating a user interface, that consider its various aspects:

- an approach based on specialized description languages;
- a context-sensitive approach;
- a data-based approach;
- an ontological approach.

The approach based on specialized description languages assumes the representation of a particular user interface in a platform-independent form. Examples of interface description languages are **UIML** [3], **UsiXML** [4], **XForms** [5] and **JavaFX FXML** [6]. The key idea of the represented languages is to build a model of dialogues and interface forms in a form independent of the used technology, a description of visual elements as well as the relations between them and their features for creating a certain user interface.

The context-sensitive approach integrates the usage of a structural description of the interface based on description

languages with a behavioral specification, that is, the generation of the interface is based on user actions. As part of the approach, transitions between different types of a particular user interface are specified. Examples of the implementation of this approach are **CAP3** [7] and **MARIA** [8].

A data-based, or a model-oriented, approach uses a model of the subject domain as the basis for creating user interfaces. Implementation includes **JANUS** [9] and **Mecano** [10].

Existing ontological approaches are usually based on the approaches presented earlier and use ontologies as a way of representation of information about a particular user interface. For example, by analogy with the approach based on specialized description languages, the framework [11] was proposed, which uses an ontology to describe the user interface based on concepts stored in the knowledge base. By analogy with the context-dependent approach, within the framework of the article [12], the model of the subject domain together with the user interface model is used, associated with the ontology of actions. The **ActiveRaUL** [13] project combines UIML with a model-oriented approach. Within the framework of this project, the ontological model of the subject domain is correlated with the ontological representation of the user interface. The approach proposed in [14] combines application data with the user interface ontology for the creation of a single description in the knowledge base for the subsequent automatic generation of various interface options for questionnaire applications with ready-made user interaction scenarios. It is also worth noting the articles [15] and [16], in which a concept is proposed, that allows combining information that is homogeneous in content into components of the interface model, liberating the interface developer from encoding, and forming information for each component of the interface model using editors controlled by the corresponding ontology models.

The principle of generating an interface based on a declarative description is the basis of a number of applied projects. For example, the **mermaid** [17] project allows automatically generating diagrams based on their description, and the **rjsf** [18] project allows generating forms for user input. In addition, a general approach to generating and displaying an interface based on its description from the server side of the application can also be found in industrial development under the terms **Server-Driven UI** or **Backend-Driven UI** [19].

The disadvantages of existing solutions for generating the user interface include the following:

- as a rule, the created models are specific to a particular platform or a certain implementation of the user interface, which hinders their reuse for other purposes;
- solutions that offer a platform-independent description allow generating only simple user interfaces that are limited in functionality (questionnaire applications, diagrams, etc.).

Among the represented approaches, the ontological one is the most preferable for the following reasons:

- it allows integrating earlier proposed approaches due to a single way of representation of knowledge;
- it allows creating the most complete description of various aspects of the user interface. The composition of this description will be discussed in more detail below;
- it simplifies the reuse of the interface by applying a single representation of the interface model for different platforms.

However, for existing solutions based on the ontological approach, the problem of compatibility of various ontologies within a unified system remains relevant as well as the lack of the ability to adapt to user requests and analyze their actions for independent improvement (as a basis for such an analysis, the ontology of user actions serves).

III. Proposed approach

Based on the conducted analysis, an approach on the ground of an ontological one is proposed, which consists in creating a complete semantic model of the interface, which will eliminate the shortcomings of existing solutions. The key features of the approach are:

- the fixation of the interface description in the form of an abstraction, regardless of the platform and device;
- the presence of a complete semantic model of the interface, that contains a “lexical” interface description (a description of the components, from which the interface is formed), a “syntactic” interface description (rules for forming a correct and full interface from its components) but also its semantic description (knowledge of which entity the displayed component a sign is). At the same time, the semantic description also includes the purpose, scope of application of the interface components and a description of the interface user activity;
- the representation of the specification of the interface generation tools and, if necessary, the tools themselves in a common format with a description of the interface through some unified knowledge representation language;
- the reduction of development costs due to the reuse of interface components;
- the reduction of development costs due to the usage of a hierarchical structuring of the user interface model, which allows independent development of components;
- universality, that is, the possibility of using the approach to build interfaces of any systems, regard-

less of their purpose. The unified principles of the building of the interface will allow the user to easily switch from using one system to another, significantly reducing the cost of training;
- the integration of the semantic interface model with other models within a unified system. For example, integration with the user model (biographical information, knowledge about the user's behavior within the system) will make the interface adaptive. In this case, an adaptive interface is understood as an interface that can adapt to a certain user or category of users (which implies not only a change in the visual component of the interface but also a change in its internal functionality).

Thus, based on the above, the following demands can be made to the technology, on the basis of which this approach can be implemented:

- the technology should provide an opportunity to describe various semantic models and their components of various types of knowledge in a common format;
- the technology should allow the simple integration of various semantic models within a unified system;
- the technology should support a component approach to creating semantic models.

Among the existing system design technologies, the *OSTIS Technology* meets the specified requirements, among the advantages of which it is also possible to additionally highlight the presence of a basic set of ontologies that can serve as the basis for the user interface model being developed.

Thus, within the framework of this approach, in the article, an option for implementing a framework for building user interfaces is proposed, which is based on the *OSTIS Technology*, which provides a universal language for the semantic representation (encoding) of information in the memory of intelligent computer systems, called an *SC-code*. Texts of the *SC-code* (sc-texts) are unified semantic networks with a basic set-theoretic interpretation. The elements of such semantic networks are called *sc-elements* (*sc-nodes* and *sc-connectors*, which, in turn, can be *sc-arcs* or *sc-edges*, depending on the directivity). The *SC-code alphabet* consists of five main elements, on the basis of which SC-code constructs of any complexity are built, as well as more particular types of sc-elements (for example, new concepts) are introduced. The memory that stores SC-code constructs is called semantic memory or *sc-memory* [20].

The architecture of each ostis-system includes a platform for interpreting semantic models of ostis-systems as well as a semantic model of the ostis-system described using the SC-code (sc-model of the ostis-system). In turn, the sc-model of the ostis-system includes the sc-model of the knowledge base, the sc-model of the problem solver and the sc-model of the interface. The principles of the structure and design of knowledge bases and problem solvers are discussed in more detail in [21] and [23], respectively. Within this article, the sc-model of the user interface will be considered, which is included in the sc-model of the interface. Its principles were described in the article [2], the development of which is this article.

The SC-code representation languages include:

- **SCg-code** – one of the possible ways of visual representation of SC-texts. The basic principle that underlies the SCg-code is that each sc-element is assigned a sc.g-element (graphical representation);
- **SCs-code** – a string (linear) version of the SC-code representation. It is designed to represent sc-graphs (SC-code texts) in the form of sequences of characters;
- **SCn-code** – a string non-linear version of the SC-code representation. The SCn-code is designed to represent sc-graphs in the form of sequences of characters formatted according to given rules, in which basic hypermedia tools, such as graphic images as well as tools of navigation between parts of sc.n-texts, can also be used [2].

Within the framework of this article, fragments of SCg- and SCn-codes [24] will be used, which are simultaneously fragments of the source texts of the knowledge base, that are understandable to both a human and to a machine. This allows making the text more structured and formalized while maintaining its readability.

IV. Problem definition

The user interface within the framework of the proposed approach is a specialized ostis-system focused on solving interface problems and that includes a knowledge base and a problem solver of the user interface. The general architecture of the ostis-system is shown in figure 1.

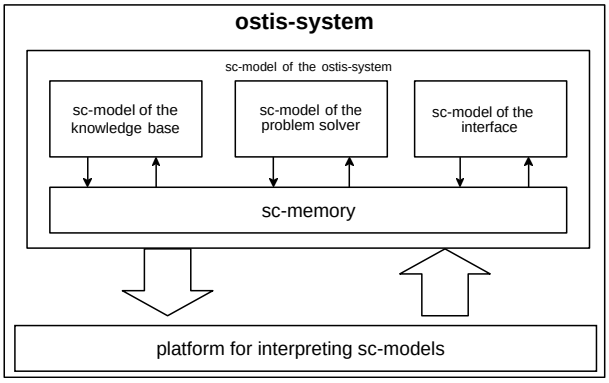

Figure 1. The architecture of the ostis-system

To solve the problem of building a user interface, the user interface knowledge base requires the presence of an sc-model of user interface components, interface actions as well as the classification of user interfaces in general,

as shown in figure 2. When designing the interface, it is proposed to use a component approach, which assumes the representation of the entire application interface in the form of separate specified components that can be developed and improved independently. It is important that, as a result of the building, the user interface should be not only static (visually formed) but also dynamic (with the ability to perform various actions, including those initiated by the user).

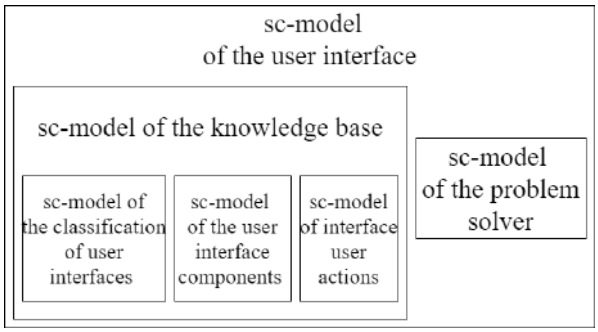

Figure 2. The structure of the sc-model of the user interface

The basis of the *sc-model of the ostis-system knowledge base* is a hierarchical system of subject domains and their corresponding ontologies. Accordingly, within the framework of the proposed approach, it is necessary to develop:

- the Subject domain of user interfaces;
- the Subject domain of user interface components;
- the Subject domain of interface user actions.

The problem solver is a hierarchical system of agents of knowledge processing in semantic memory (sc-agents), which interact with each other exclusively by specifying the actions they perform in the memory. An sc-agent is a certain subject that can perform actions in sc-memory, which belong to a certain class of autonomous actions. An autonomous action is an action that is performed regardless of whether the specified action is part of the decomposition of a more common action [22]. To build a user interface, it is necessary to implement the following agents:

- the Agent of interpretation of the sc-model of the user interface knowledge base;
- the Agent of processing of user actions.

V. Sc-model of the knowledge base

A. Subject domain of user interfaces

The subject domain of user interfaces includes the classification of user interfaces.

user interface
⊃ *command-line interface*
⊃ *graphical user interface*
  ⊃ *WIMP-interface*
⊃ *SILK-interface*
  ≔ [(Speech – речь, Image – образ, Language – язык, Knowledge – знание)]
  ⊃ *natural-language interface*
    ⊃ *speech interface*

A *user interface* is one of the most important components of a computer system. It is a set of hardware and software tools that provide information exchange between the user and the computer system.

A *command-line interface* is a user interface, in which information is exchanged between a computer system and a user by writing text instructions or commands.

A *graphical user interface* is a user interface, in which information is exchanged between a computer system and a user using the graphical components of a computer system.

A *WIMP-interface* is a user interface, in which information is exchanged between a computer system and a user in the form of a dialogue using windows, menus and other controls.

A *SILK-interface* is a user interface that is closest to the natural form of human communication. The computer system initiates commands independently, analyzing human speech and finding key phrases in it. The result of running commands is converted into a form that is understandable to a human, for example, into a natural-language form or an image.

A *natural-language interface* is a SILK-interface, in which the exchange of information between a computer system and a user occurs through a dialogue. The dialogue is conducted in one of the natural languages.

A *speech interface* is a SILK-interface, in which information is exchanged through a dialogue, during which the computer system and the user communicate using speech. This type of interface is the closest to natural communication between humans.

B. Subject domain of user interface components

The subject domain of user interface components describes the structure and features of the visual representation of user interface components. The Ui2Ont ontology [26] was taken as the basis of this subject domain.

A *user interface component* is a sign of a fragment of the knowledge base, that has a certain form of external representation on the screen.

user interface component
⇒ *decomposition**:
  {• *atomic user interface component*
  • *non-atomic user interface component*
  }

An *atomic user interface component* is a user interface component that does not contain other user interface components in its structure.

A *non-atomic user interface component* is a user interface component that consists of other user interface components.

Below is the classification of the components:

user interface component
- ⊃ *presentation user interface component*
  - ⊃ *output*
    - ⊃ *image-output*
    - ⊃ *graphical-output*
      - ⊃ *chart*
      - ⊃ *map*
      - ⊃ *progress-bar*
    - ⊃ *video-output*
    - ⊃ *sound-output*
    - ⊃ *text-output*
      - ⊃ *headline*
      - ⊃ *paragraph*
      - ⊃ *message*
  - ⊃ *decorative user interface component*
    - ⊃ *separator*
    - ⊃ *blank-space*
  - ⊃ *container*
    - ⊃ *menu*
    - ⊃ *menu-bar*
    - ⊃ *tool-bar*
    - ⊃ *status-bar*
    - ⊃ *table-row-container*
    - ⊃ *list-container*
    - ⊃ *table-cell-container*
    - ⊃ *tree-container*
    - ⊃ *labeled-group*
    - ⊃ *tab-pane*
    - ⊃ *spin-pane*
    - ⊃ *tree-node-container*
    - ⊃ *scroll-pane*
    - ⊃ *window*
      - ⊃ *modal-window*
      - ⊃ *non-modal-window*
- ⊃ *interactive user interface component*
  - ⊃ *data-input-component*
    - ⊃ *data-input-component-with-direct-feedback*
      - ⊃ *text-input-component-with-direct-feedback*
        - ⊃ *multi-line-text-field*
        - ⊃ *single-line-text-field*
      - ⊃ *slider*
      - ⊃ *drawing-area*
      - ⊃ *selection-component*
        - ⊃ *selection-component-multiple-values*
        - ⊃ *selection-component-single-values*
      - ⊃ *selectable-data-representation*
        - ⊃ *check-box*
        - ⊃ *radio-button*
        - ⊃ *toggle-button*
        - ⊃ *selectable-item*
    - ⊃ *data-input-component-without-direct-feedback*
      - ⊃ *spin-button*
      - ⊃ *speech-input*
      - ⊃ *motion-input*
  - ⊃ *presentation-manipulation-component*
    - ⊃ *activating-component*
    - ⊃ *continuous-manipulation-component*
      - ⊃ *scrollbar*
      - ⊃ *resizer*
  - ⊃ *operation-trigger-component*
    - ⊃ *command-selection-component*
      - ⊃ *button*
      - ⊃ *menu-item*
    - ⊃ *command-input-component*

A *presentation user interface component* is a component of the user interface that does not imply interaction with the user.

A *decorative user interface component* is a user interface component designed to style the interface.

A *container* is a user interface component, whose task is to place a set of components included in its structure.

A *window* is a separate screen panel that contains various elements of the user interface. Windows can be placed on top of each other.

A *modal-window* is a window that blocks the user experience with the application until the user closes the window.

A *non-modal-window* is a window that allows the user to interact with other windows without having to close this window.

An *interactive user interface component* is a user interface component that is used to interact with the user.

A *data-input-component* is a user interface component designed for input of information.

A *presentation-manipulation-component* is a user interface component designed to represent information and interact with the user.

An *operation-trigger-component* is a user interface component that requests the user to perform some action.

A non-atomic component is connected to its constituent components using the *decomposition** relation. Here is an example of a description of a non-atomic component of the main window. The appearance of the display of this component is shown in figure 3.

The formalization of this component in the SCn-language looks like in the following manner:

MainPage
- ∈ *window*
- ⇒ *decomposition**:
  - {• *Navigation*
    - ∈ *non-atomic user interface component*
    - ⇒ *decomposition**:

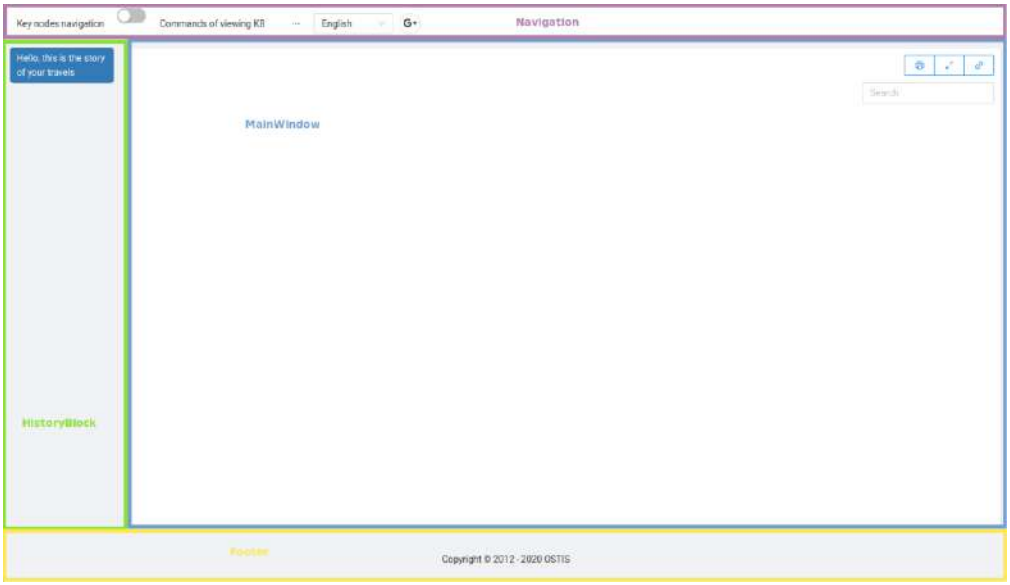

Figure 3. An example of the display of a non-atomic component

{• *mainMenu*
∈ *menu*
⇒ *decomposition**:
{• *item1*
∈ *menu-item*
• *item2*
∈ *menu-item*
• *switch*
∈ *toggle-button*
}
• *languageSelect*
∈ *selection-component-single-values*
• *googleAuth*
∈ *button*
}

- *HistoryBlock*
  ∈ *non-atomic user interface component*
- *MainBlock*
  ∈ *non-atomic user interface component*
  ⇒ *decomposition**:
  {• *mainWindow*
  ∈ *window*
  • *tool-bar*
  ⇒ *decomposition**:
  {• *printButton*
  ∈ *button*
  • *resizeButton*
  ∈ *button*
  • *linkButton*
  ∈ *button*
  • *searchInput*
  ∈ *single-line-text-field*
  }
  }
  • *Footer*
  ∈ *non-atomic user interface component*
  }

The subject domain of user interface components also contains a description of the properties of the components.

As part of the work on the knowledge base of the *IMS.ostis Metasystem* [24], the subject domain of spatial entities and their forms was created. The *IMS.ostis Metasystem* is an intelligent metasystem built according to the standards of the OSTIS Technology and aimed at usage by ostis-system engineers – at supporting the design, implementation and updating (reengineering) of ostis-systems – and at developers of the OSTIS Technology – at supporting collective activities for the development of standards and libraries of the OSTIS Technology. In the subject domain of spatial entities and their forms, there are descriptions of such concepts as:

- spatial entity;
- form;
- coordinate system;
- Cartesian coordinate system;
- two-dimensional Cartesian coordinate system;
- point of reference;
- point;
- segment;
- length;
- thickness;
- height;
- width;
- rectangle.

The subject domain of user interface components inter-

sects with the subject domain of spatial entities and their forms and adds a set of concepts to describe the properties of components, part of which is given below.

*Text** is a binary relation that connects a user interface component to a file that contains the text of the user interface component.

Color is a parameter of the user interface component that determines its color.

Text color is a parameter of the user interface component that determines the color of its text.

Text size is a parameter of the user interface component that determines the size of its text.

Text font is a parameter of the user interface component that determines the font of its text.

The *deactivation property* is a logical parameter of a user interface component that can be set to inhibit the usage of the component until a certain action is performed.

The *maximum number of characters* is a parameter of the text-input-component-with-direct-feedback component, which sets the maximum number of characters that can be input by the user.

Thus, within the framework of this subject domain, both component classes and their instances are described as well as the properties of components for visualization, regardless of the platform. At the same time, these components and properties are easily changeable and extensible.

C. Subject domain of interface user actions

The Subject domain of interface user actions contains a specification of user actions, which can be performed for the components of the user interface. The Ui2Ont ontology [26] was used as the basis of this subject domain.

An *interface user action* is a minimally meaningful fragment of some activity of the user, performed through the interface.

Next is the classification of interface user actions.

interface user action

- ⊃ *mouse-action*
  - ⊃ *mouse-scroll*
    - ⊃ *mouse-scroll-up*
    - ⊃ *mouse-scroll-down*
  - ⊃ *mouse-hover*
  - ⊃ *mouse-drop*
  - ⊃ *mouse-click*
    - ⊃ *mouse-double-click*
    - ⊃ *mouse-single-click*
  - ⊃ *mouse-gesture*
  - ⊃ *mouse-unhover*
  - ⊃ *mouse-drag*
- ⊃ *speech-action*
- ⊃ *keyboard-action*
  - ⊃ *press-function-key*
  - ⊃ *type-text*
- ⊃ *tangible-action*
- ⊃ *touch-action*
  - ⊃ *touch-click*
    - ⊃ *touch-single-click*
    - ⊃ *touch-double-click*
  - ⊃ *touch-gesture*
    - ⊃ *one-fingure-gesture*
    - ⊃ *multiple-finger-gesture*
  - ⊃ *touch-drop*
  - ⊃ *touch-drag*
- ⊃ *pen-base-action*
  - ⊃ *touch-function-key*
  - ⊃ *draw*
  - ⊃ *write-text*

A *mouse-hover* is the interface user action, which corresponds to the appearance of the mouse cursor within the user interface component.

A *mouse-drop* is the interface user action, which corresponds to dropping some component of the user interface within another user interface component using the mouse.

A *mouse-gesture* is an interface user action, which corresponds to the performance of a certain gesture through the movement of the mouse.

A *mouse-unhover* is an interface user action, which corresponds to the exit of the mouse cursor outside the framework of the user interface component.

A *mouse-drag* is an interface user action, which corresponds to dragging a user interface component with the mouse.

A *tangible-action* is an interface user action performed using taction.

A *touch-action* is an interface user action performed using the sensor.

A *touch-gesture* is an interface user action, which corresponds to the performance of a certain gesture with the movement of fingers on the screen of the sensor.

A *one-fingure-gesture* is an interface user action, which corresponds to the performance of a certain gesture by moving one finger on the screen of the sensor.

A *multiple-fingure-gesture* is an interface user action, which corresponds to the performance of a certain gesture by moving several fingers on the screen of the sensor.

A *touch-drop* is an interface user action, which corresponds to dropping a certain component of the user interface within another component of the user interface using a sensor.

A *touch-drag* is an interface user action, which corresponds to dragging a certain component of the user interface using a sensor.

A *pen-base-action* is an interface user action performed using a pen on a graphics tablet.

A *touch-function-key* is an interface user action, which corresponds to pressing the function key of the graphic tablet with a pen.

The above user actions are common to all systems. It should be noted that the interface user action, as a rule, initiates some internal action of the system.

internal action of the system
⊃ *internal action of the ostis-system*

In the case of ostis-systems, as part of the work on the knowledge base of the IMS.ostis Metasystem [24], the *Subject domain and ontology of actions, problems, plans, protocols and methods implemented by the ostis-system in its memory as well as internal agents that perform these actions* was allocated. A fragment of this subject domain is shown below.

internal action of the ostis-system
:= [an action in sc-memory]
:= [an action performed in sc-memory]

Each *internal action of the ostis-system* denotes some transformation performed by some *sc-agent* (or a group of *sc-agents*) and focused on the transformation of *sc-memory*.

action in sc-memory
⊃ *action in sc-memory initiated by a question*
⊃ *action of editing the ostis-system knowledge base*
⊃ *action of setting the mode of the ostis-system*
⊃ *action of editing a file stored in sc-memory*
⊃ *action of interpreting a program stored in sc-memory*

An *action in sc-memory initiated by a question* is an action aimed at forming an answer to the raised question.

To define an action that is initiated when interacting with the user interface, the *action initiated by the user interface** relation is used.

action initiated by the user interface*
∈ *quasi-binary relation*
∈ *oriented relation*
⇒ *first domain**:
user interface component ∪ *user interface action class*
⇒ *second domain**:
class of internal actions of the system

The first component of the binding of the *action initiated by the user interface** relation is a binding, the elements of which are an element of the set of user interface components and an element of the *user interface action class* set. The second component is an element of the *class of internal actions of the system* set. An example of using this relation is shown in figure 4.

Thus, within the framework of these subject domains, interface user actions and internal actions of the system are described. These actions are basic and can be easily expanded and refined.

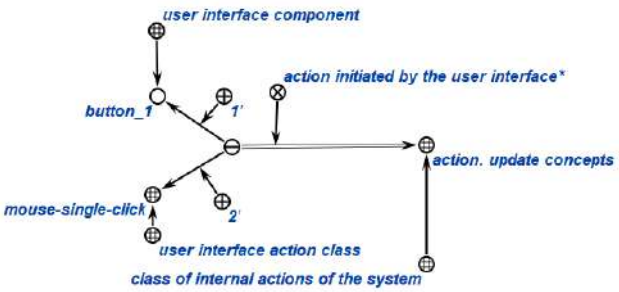

Figure 4. An example of using the action initiated by the user interface* relation

The integration of the abovementioned ontologies allows implementing an approach, within the framework of which:

- all the components of the user interface correspond to a certain fragment of the knowledge base. It allows addressing various questions about these components to the system. As examples of such questions, the following ones can act: “what class of components does the specified component belong to?”, “what is the specified component designed for?”, “what does the specified component consist of”, etc.;
- the classification of components allows the further building of the user interface taking into account the knowledge about them. For example, the presentation user interface component can be highlighted in one color and the interactive user interface component – in another;
- it is possible to accumulate the results of the interface user activity to further adapt the interface for them. Changing the interface is reduced to changing its sc-model, which can be carried out on the basis of logical rules, which are also described in the system knowledge base. For example, the system may contain a logical rule for adding the most frequently initiated interface user actions to a separate component;
- it is possible to analyze the efficiency of user actions for further improvement of the interface (for example, several interface actions performed by the user in a row can be replaced by one);
- the user will have the opportunity to receive answers to questions about the organization of interface activities. Examples of such questions include: “what interface actions can be performed for the specified component?”, “what interface actions were performed the most often?”, etc.

VI. Sc-model of the problem solver

From the point of view of processing the sc-model of the user interface knowledge base, the following problems should be solved:

- the interpretation of the sc-model of the user interface knowledge base (building the user interface);
- the processing of user actions.

User interface problem solver
⇐ *decomposition of an abstract sc-agent**:
{• *Agent of interpretation of the sc-model of the user interface knowledge base*
• *Agent of processing of user actions*
}

An *Agent of interpretation of the sc-model of the user interface knowledge base* accepts an instance of the user interface component for displaying as an input parameter. In this case, the component can be either atomic or non-atomic (for example, a component of the main application window). The result of the operation of the agent is a graphical representation of the indicated component, taking into account the used implementation of the platform for interpreting semantic models of ostis-systems.

The operation algorithm of this agent is as follows:

- the input component type (atomic or non-atomic) is checked;
- if the component is atomic, then to display its graphical representation based on the properties specified for it. If this component is not included in the decomposition of any other component, to complete the performance. Otherwise, to determine the component, the decomposition of which includes the considered component, apply its properties to the current atomic component and start processing the found non-atomic component, going to the first item;
- if the component is non-atomic, then to check whether the components, into which it was decomposed, were displayed. If yes, then to complete the performance, otherwise to determine the component from the decomposition of the non-atomic component being processed, which has not yet been displayed, and start processing the found component by going to the first item.

An *agent of processing of user actions* is a non-atomic agent that includes many agents, each of which processes user actions of a certain class (for example, an agent of processing a mouse click action, an agent of processing a mouse drop action, etc.). The agent reacts to the appearance of an instance of an interface user action in the knowledge base of the system, finds an internal action class associated with it and generates an instance of this internal action for subsequent processing.

VII. Implementation of the proposed approach

The current implementation of the sc-model interpretation platform is web-oriented [27].

Taking into account the features of the platform and for the possibility of integrating the proposed approach with existing solutions in the field of building user interfaces, it is proposed to implement the agent of interpretation of the sc-model of the user interface knowledge base as a non-atomic agent that is decomposed into the agent of translation of the sc-model of the user interface knowledge base into a format compatible with existing solutions and the agent of displaying the specified format in the graphical representation of the user interface.

It is proposed to use the JSON format as an intermediate description format for a number of reasons:

- it is the most popular format for data transmission and storage in modern systems;
- the compact and simple syntax;
- the simplicity of making changes;
- the simplicity of transmission and processing.

Thus, an additional *agent of translation of the description of the user interface component from the sc-model to the JSON format* is introduced. As an input parameter, this agent accepts an instance of the translation user interface component in the JSON format. The JSON description is formed by recursively processing the description of components from non-atomic to atomic ones.

The *Agent of displaying the specified format in the graphical representation of the user interface* is non-atomic and is decomposed into a set of agents that perform displaying for various interpretation platforms (web, mobile, desktop computer platforms, etc.). As input, this agent accepts a description of the user interface component in the JSON format. The result of the operation of the agent is a graphical representation of the user interface.

For the possibility of changing the sc-model of the user interface, editing tools such as the SCg-, SCs- and SCn-editors are implemented within the OSTIS Technology. So, the framework proposed within the approach includes three key parts:

- an sc-model of the user interface;
- tools of visualization of the sc-model of the user interface;
- tools of editing of the sc-model of the user interface.

The general structure of the framework is shown in figure 5.

VIII. Technique of developing user interface components

One of the advantages of the proposed approach is the accumulation of frequently used components. To do this, it is supposed to create a library of components with preset properties, which is included in the subject domain of user interface components. The components included in the library are platform-independent (they can be visualized regardless of the used interpretation platform).

The process of creating an instance of a user interface component can be described as follows:

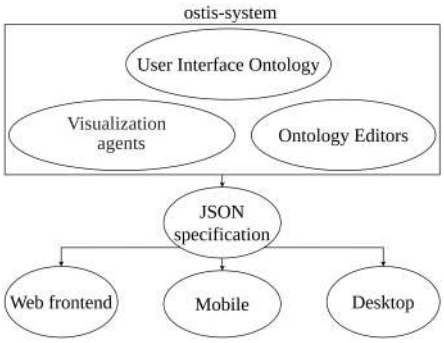

Figure 5. The structure of the framework for generating the user interface

- to check whether the class of the necessary component is present in the subject domain of the user interface components;
- if it is not available, it is necessary to create a class of the necessary component, specifying the set of properties for it;
- to check whether there is a description of the instance of the required component in the component library;
- if it is missing, it is required to supplement the component library with a description of a new instance of the component with preset properties;
- if necessary, to create a new instance of the class of the necessary component, setting it certain properties and actions based on the ontology of the subject domain of a particular system;
- to run the agent of visualization of the component instance for the used interpretation platform.

The described process is shown in figure 6.

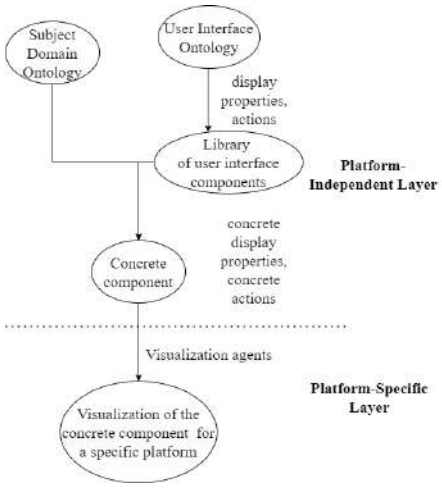

Figure 6. The process of creating an instance of a user interface component

IX. Examples of the framework operation

Next, we will give some examples of the description of interface components in the knowledge base of the system in the SCg-language and the result of their visualization. Figure 7 shows the formalization of the "button" component, figure 8 shows the result of its display in the web interface. In figures 9, 11, the description of the "text field" and "form" components is presented, in figures 10, 12 is their display, respectively.

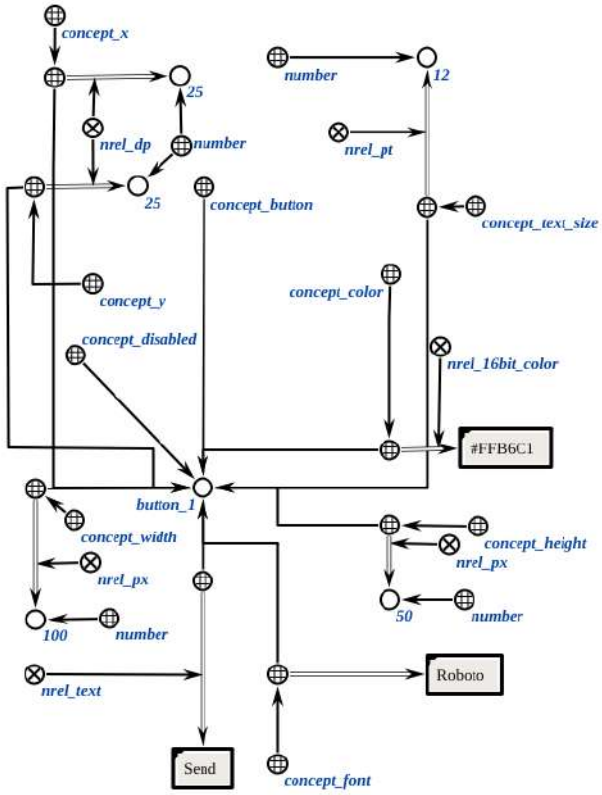

Figure 7. The formalization of the "button" component

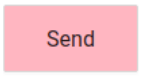

Figure 8. The result of the display of the "button" component in the interface

X. Conclusion

In this article, an ontological approach to the building of semantic models of user interfaces based on the OSTIS Technology is proposed.

The analysis of existing approaches to the building of user interfaces is carried out, the structure and technique of building user interface components for the framework proposed within the approach are presented. Examples of the ontological description of the interface components

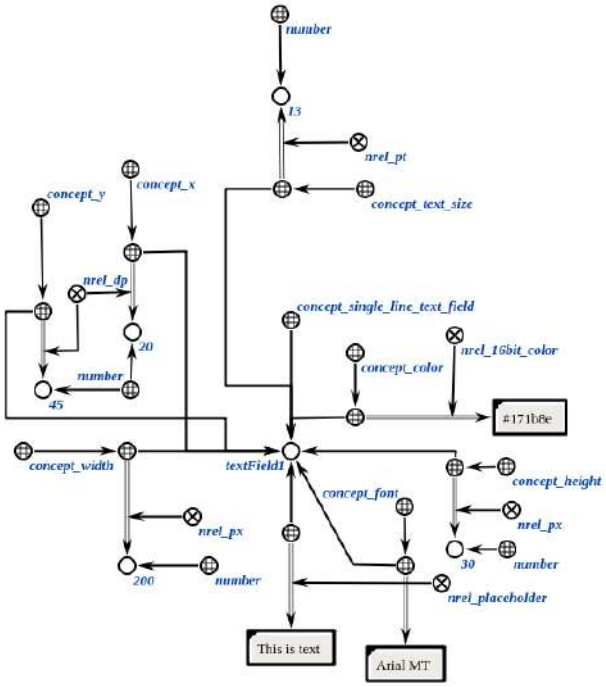

Figure 9. The formalization of the "text field" component

Figure 10. The result of the display of the "text field" component in the interface

and the results of their visualization using the developed tools for the automatic building are also presented.

In contrast to the existing approaches, the proposed approach will allow:

- taking into account the semantics of the user interface components when building it;
- generating questions to the system related to the user interface;
- taking into account the history of the interface user activity to improve the quality of their work with the system;
- rebuilding the user interface by changing its model during the operation of the system.

At this stage, according to the proposed approach, the following were implemented:

- a fragment of the subject domain of user interface components;
- a fragment of the subject domain of interface user actions;
- an agent of translation of the description of the user interface component from the sc-model to the JSON format;
- an agent of displaying the JSON format in a graphical representation of the user interface for a web platform.

As part of further work, it is planned to expand specified subject domains and implement visualization agents for other platforms.

Acknowledgment

The author would like to thank the Department of Intelligent Information Technologies of the Belarusian State University of Informatics and Radioelectronics for the help and valuable comments.

References

[1] Data-driven UI: unlimited power [Electronic resourse]. Access mode: https://mobius-piter.ru/en/2018/spb/talks/v96lokugwe8cwggio8ois/ Date of access: 21.05.2021.

[2] Boriskin A. S., Sadouski M. E., Koronchik D. N., Zhukau I. I., Khusainov A. F. Ontology-Based Design of Intelligent Systems User Interface. Otkrytye semanticheskie tekhnologii proektirovaniya intellektual'nykh system [Open semantic technologies for intelligent systems], Minsk, 2017, pp. 95–106

[3] Abrams M., Phanouriou C., Batongbacal A. L., Williams S. M., Shuster J. E. UIML: An appliance-independent XML user interface language. 99 Proceedings of the eighth international conference on World Wide Web, 1999, pp. 1695–1708

[4] Limbourg Q. USIXML: A User Interface Description Language Supporting Multiple Levels of Independence. Matera, M. and Comai, S. (eds.) ICWE Workshops, Rinton Press, 2004, pp. 325–338

[5] XForms 1.1 [Electronic resourse]. Access mode: https://www.w3.org/TR/xforms Date of access: 25.05.2021.

[6] Introduction to FXML [Electronic resourse]. Access mode: https://openjfx.io/javadoc/12/javafx.fxml/javafx/fxml/doc-files/introduction_to_fxml.html Date of access: 23.05.2021.

[7] Van den Bergh J., Luyten K., Coninx K. CAP3: Context-Sensitive Abstract User Interface Specification. Proceedings of the 3rd ACM SIGCHI symposium on Engineering interactive computing systems – EICS'11, 2011, pp. 31–40

[8] Paterno F., Santoro C., Spano L.D. Maria: A Universal, Declarative, Multiple Abstraction-Level Language for Service-Oriented Applications in Ubiquitous Environment. ACM Trans. Comput. Interact. 16, 2009.

[9] Balzert H., Hofmann F., Kruschinski V. The JANUS Application Development Environment—Generating More than the User Interface. Computer Aided Design of User Interfaces, Vol. 96, 1996, pp. 183–206

[10] Puerta A.R., Eriksson H., Gennari J.H., Musen M.A. Beyond data models for automated user interface generation. In Proceedings British HCI'94, 1994.

[11] Liu B., Chen H., He W. Deriving user interface from ontologies: A model-based approach. Int. Conf. Tools with Artif. Intell. ICTAI, 2005, pp. 254–259

[12] Gaulke W., Ziegler J. Using profiled ontologies to leverage model driven user interface generation. 7th ACM SIGCHI Symp. Eng. Interact. Comput. Syst. – EICS '15, 2015, pp. 254–259

[13] Sahar A., Armin B., Shepherd H., Lexing L. ActiveRaUL : Automatically generated Web Interfaces for creating RDF data, 2013.

[14] Michael Hitz, Thomas Kessel Using Application Ontologies for the Automatic Generation of User Interfaces for Dialog-Based Applications. Research and Practical Issues of Enterprise Information Systems, CONFENIS 2016

[15] Gribova V. V., Cherkezishvili N. N. Avtomatizaciya razrabotki pol'zovatel'skih interfeysov s dinamicheskimi dannymi [Automating the development of user interfaces with dynamic data]. Otkrytye semanticheskie tekhnologii proektirovaniya intellektual'nykh system [Open semantic technologies for intelligent systems], 2011, pp. 287–292 (in Russian)

[16] Gribova V.V, Tarasov A.V. Generator koda pol'zovatel'skogo interfejsa, upravlyaemyy ontologiey. Artificial Intelligence, Vol. 4, 2005, pp. 457 – 464 (in Russian)

[17] Mermaid documentation [Electronic resourse]. Access mode: https://mermaid-js.github.io/mermaid Date of access: 18.04.2021.

[18] React Json Schema Form [Electronic resourse]. Access mode: https://rjsf-team.github.io/react-jsonschema-form/ Date of access: 20.04.2021.

[19] Exploring Server-Driven UI [Electronic resourse]. Access mode: https://betterprogramming.pub/exploring-server-driven-ui-cf67b3da919 Date of access: 22.04.2021.

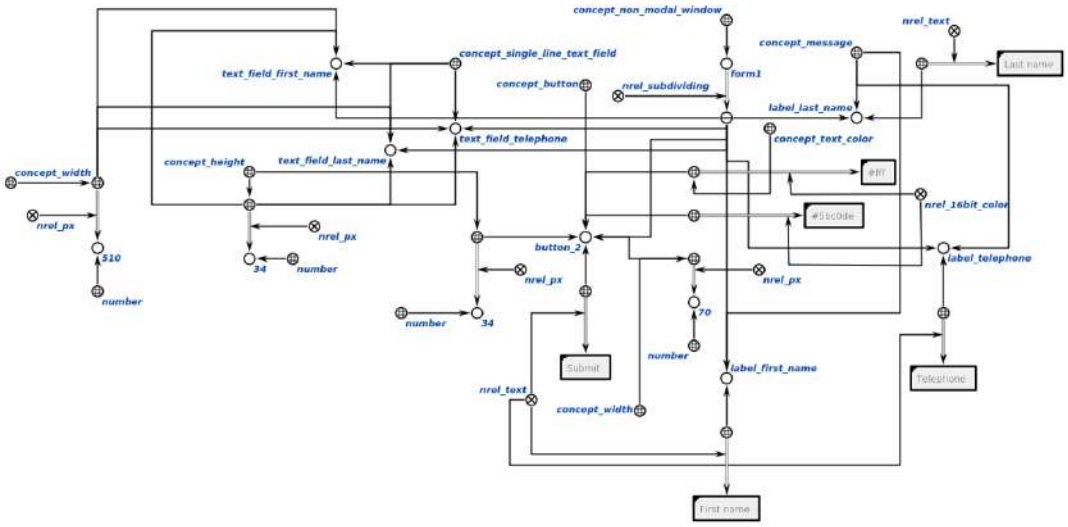

Figure 11. The formalization of the “form” component

Figure 12. The result of the display of the “form” component in the interface

[20] Golenkov V., Guliakina N., Davydenko I., Eremeev A. Methods and tools for ensuring compatibility of computer systems. Otkrytye semanticheskie tekhnologii proektirovaniya intellektual’nykh system [Open semantic technologies for intelligent systems], 2019, pp. 25–52

[21] Davydenko I. Semantic models, method and tools of knowledge bases coordinated development based on reusable components. Otkrytye semanticheskie tekhnologii proektirovaniya intellektual’nykh system [Open semantic technologies for intelligent systems], 2018, pp. 99–118

[22] Shunkevich D. Ontology-based Design of Knowledge Processing Machines. Otkrytye semanticheskie tekhnologii proektirovaniya intellektual’nykh system [Open semantic technologies for intelligent systems], 2017, pp. 73–94

[23] Shunkevich D. Agent-oriented models, method and tools of compatible problem solvers development for intelligent systems. Otkrytye semanticheskie tekhnologii proektirovaniya intellektual’nykh system [Open semantic technologies for intelligent systems], 2018, pp. 119–132

[24] IMS.ostis Metasystem [Electronic resourse]. Access mode: http://ims.ostis.net/ Date of access: 03.06.2021.

[25] Golenkov V. [et al.] Principles of organization and automation of the semantic computer systems development. Otkrytye semanticheskie tekhnologii proektirovaniya intellektual’nykh system [Open semantic technologies for intelligent systems], 2019, pp. 53-90

[26] Paulheim H., Probst F. UI2Ont – A Formal Ontology on User Interfaces and Interactions. Semantic Models for Adaptive Interactive Systems, 2013, pp. 1–24

[27] OSTIS Web Platform [Electronic resourse]. Access mode: https://github.com/ostis-dev/ostis-web-platform Date of access: 25.04.2021.

Онтологический подход к построению семантических моделей пользовательских интерфейсов

Садовский М. Е.

В работе предлагается онтологический подход к построению семантических моделей пользовательских интерфейсов на основе Технологии OSTIS.

Проведен анализ существующих подходов к построению пользовательских интерфейсов, приведена структура и представлена методика создания компонентов пользовательского интерфейса для предлагаемого в рамках подхода фреймворка. Также представлены примеры онтологического описания компонентов интерфейса и результаты их визуализации с помощью разработанных средств автоматического построения.

В отличие от существующих подходов предлагаемый подход позволит:

- учитывать семантику компонентов пользовательского интерфейса при его построении;
- формировать вопросы к системе, связанные с пользовательским интерфейсом;
- учитывать историю интерфейсной деятельности пользователя для повышения качества его работы с системой;
- перестраивать пользовательский интерфейс путем изменения его модели в процессе работы системы.

На данном этапе согласно предлагаемому подходу были реализованы:

- фрагмент предметной области компонентов пользовательских интерфейсов;
- фрагмент предметной области интерфейсных действий пользователя;
- агент трансляции описания компонента пользовательского интерфейса из sc-модели в JSON формат;
- агент отображения JSON формата в графическое представление пользовательского интерфейса для web-платформы.

В рамках дальнейшей работы планируется расширение указанных предметных областей и реализация агентов визуализации для других платформ.

Received 01.06.2021