account by the developer at the time of creation of the computer system and are laid in its program code in the form of ready-made structures created under the influence of the model of communication with the user, the model of the environment, the model of the user and the proposed algorithm for solving the problem. Often this modeling process does not lead to an optimal result. In addition, computer systems tend to receive new functions within their life cycle, which entails updating the user interface, which in turn leads to repeated manual design of the user interface taking into account all the factors described above.

User interfaces of computer systems are suboptimal largely because the toolkit for creating user interfaces with tools to solve the problems described above is not sufficiently refined, versatile, or widespread. In order to create a complete interface development toolkit, it is necessary to describe the mechanisms of system-human communication, in particular, to define the classes of transmitted messages in order to build a model of human communication based on the task facing the system. It is also necessary to describe what components of the user interface implement certain messages between the system and the user; describe relevant properties of the user and the environment, and automate the rules by which the user interface should change in the presence of certain properties of the environment or the user. Thus, formalize the experience of experts in building user interfaces and turn it into an automated algorithm. Such a tool would have the ability to generate an interface for systems with dynamic problem formulation, adapt and personalize the interface to the user and the environment.

III. Proposed approach

A. General principles of building adaptive user interfaces of intelligent systems.

An ontological approach based on the semantic model proposed in [3] is suggested to build adaptive user interfaces of intelligent systems. This approach assumes:

- "lexical" interface description — a description of the components from which the interface is formed;
- "syntactic" description of an interface — rules for forming a correct interface from its components;
- semantic description. Knowledge about what entity the displayed component is a sign of. The semantic description also includes the purpose, scope of application of the interface components, description of the user's interface activity, etc. The semantic description also includes the purpose, scope of application of the interface components, description of the user's interface activity, etc.

The following ontologies need to be implemented within the approach:

- ontology of problems and algorithms for their solution;
- ontology of user interfaces of intelligent computer systems;
- ontology of components of user interfaces of intelligent computer systems;
- ontology of the context of use of user interfaces of intelligent computer systems;
  - ontology of users of intelligent computer systems;
  - ontology of intelligent computer systems users' actions;
  - ontology of the environment of intelligent computer systems;
  - ontology of devices of intelligent computer systems;
- ontology of external information constructs;
- ontology of incoming and outgoing messages of intelligent computer systems.

In addition to the proposed ontologies, it is necessary to develop tools for automatic interface generation and editing of its model to enable interface modification during operation and tools to support the design of user interfaces.

The above means are also proposed to be realized with the help of the above ontologies, which allow to fully describe in the knowledge base of the intelligent system both the interface itself and the principles of its interaction with the user, as well as the mechanisms of adaptation of the interface depending on the user and the environment. It is important to note that algorithms for solving user tasks are also proposed to be formed dynamically in the knowledge base of the system.

The application of this approach requires a basis in the form of intelligent systems design technology, which allows solving complex problems and whose components integrate with each other and have a synergistic effect. In turn, the application of the ontological method of building adaptive user interfaces within the framework of such technology will make intelligent systems practically applicable for a large number of classes of tasks.

The technology that meets these requirements is the OSTIS Technology. Intelligent systems developed according to the OSTIS Technology are called ostis-systems [10].

The advantages of the OSTIS Technology are as follows:

- unification of different types of knowledge with the help of SC-code;
- ontological approach to knowledge structuring;
- availability of a variant of realization of the platform of interpretation of semantic models of intellectual systems (sc-models);
- problem solver is based on a multi-agent approach in which agents interact with each other solely by specifying the actions they perform in a shared semantic memory;

- library of reusable knowledge base components and problem solvers;
- the possibility of semantic representation of logical formulas and statements;
- availability of a variant of implementation of the interpreter of logical models of problem solving.

The OSTIS Technology allows to implement the user interface in the simplest and most efficient way due to the use of automation tools for the design of user interface components, as well as due to the library of reusable components of ostis-systems, which significantly reduces the time of interface development [11].

The main ostis-system within the OSTIS Ecosystem is the OSTIS Metasystem [12]. The OSTIS Metasystem allows to automate the development of ostis-systems and their components, including user interfaces [13].

The internal universal language of knowledge representation is SC-code [14]. With the help of SC-code the interface of ostis-systems itself and the programs and rules that are used when using the interface are recorded in a unified way. That is, using SC-code, it is possible to write the interface, and also the program which can not only implement some business logic of system, but also the program which will change the interface itself which is used at interaction with the user. Through the use of SC-code and the principles of the OSTIS Technology it is possible to realize the adaptability of user interfaces in the simplest and most flexible way.

To implement the user interface as a subsystem with its own knowledge base and problem solver, it is necessary to develop the listed family of user interface ontologies, as well as a collective of agents for the functioning of the user interface:

- non-atomic sc-agent of interpreting sc-model of user interfaces of ostis-systems;
- non-atomic sc-agent of generation of sc-model of user interface of ostis-systems;
- non-atomic sc-agent of adaptation of sc-model of user interface of ostis-systems to a particular user and external environment;
- non-atomic sc-agent of interpretation of user actions; user interfaces of ostis-systems.

B. User interface module for ostis-systems

The implementation of the user interface of ostis-systems is based on:

- extensibility without the need to change the logic of the module (for personalization and adaptation on the part of the user and for styling and fine-tuning on the part of the system developer);
- maximum simplification for developers to add a new user interface platform (other than the current weboriented);
- ensuring the simplest possible integration of the UI module into existing systems;
- ensuring that the final interface is comparable in speed to user interfaces developed by traditional means;
- support for a high level of abstraction for operating user interface objects, thus not limiting developers using any means to communicate with users (including virtual reality helmets or interaction with real world objects).

In this regard, the following architectural decisions have been made:

- to implement the functionality of translating the scmodel of the user interface into a specific platformdependent markup on the side of the ostis-systems problem solver;
- to implement the functionality of user actions interpretation on the side of ostis-systems problem solver;
- to implement a mechanism to track updates to the user interface model and pass only the changed part of the markup to the user interface implementation platform;
- to use a web-oriented platform as the first supported platform for user interface implementation;
- not to use web primitives to describe interfaces, but to limit ourselves to higher-level concepts and relations, such as size, algorithm of mutual arrangement, decomposition of a user interface element (an example of the description of the "Button" component is presented in Figure 1);
- to provide the possibility of applying the rules of user interface adaptation defined declaratively in the knowledge base of ostis-systems.

The general architecture of the module is presented in Figure 2.

The order of interaction can be described as follows:

- the user interacts with the user interface of the ostissystem to solve the task he/she needs;
- information about the interaction is transferred to the knowledge base of the system;
- initiation of a team of agents to solve the user's task is performed;
- results of the user's task solution are stored in the system's knowledge base;
- the user interface agent collective is initiated;
- user interface agents modify the interface model in the knowledge base based on the received information about the result of the performed task as well as the current usage context;
- as a result, adaptation is performed - the user interface model in the knowledge base of the system is changed;
- changing the user interface model in the knowledge base of the system leads to an updated visualization of the user interface.

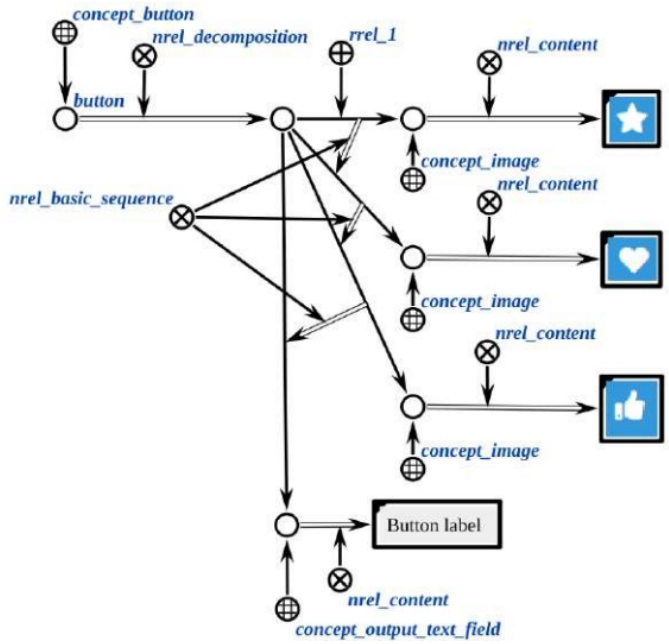

Figure 1: Example of button decomposition

Description of user's action takes place in the knowl edge base of the system. An example of formalization of a single button press is shown in Figure 3. The sequence of click-based UI updates is shown in Figure 4.

The algorithm for updating the UI after a user action involves the following steps:

1) The user performs the action of pressing the button.
2) The user interface captures information about the user's action in the system's knowledge base.
3) When this information appears, the user action interpretation agent (UIActionAgent) is initiated.
4) This agent searches the knowledge base for an internal system action that should be executed as a result of pressing a button. The search is based on information about the class of the user action and the UI component for which it was initiated.
5) Having received a template for creating an internal action in the system's knowledge base, UiActionA gent initiates its execution.
6) An internal system action is performed. If necessary, the user interface model in the knowledge base is changed.
7) An UpdateModelAgent call is executed to update the user interface model.
8) For each modified component of the user interface, the agent of forming a visualization of this compo nent for the used platform is called.
9) The modified part of the user interface is redrawn. The key components of the module are:

- User interface sc-model generator. It is responsible for compiling requirements for the current state of the user interface based on the solution step of the current task. Based on the solution algorithm, the generator finds a set of input information received from the user and a set of elements on which it is necessary to give feedback. Further, the genera tor specifies the types of received information and feedback, searches for the most general classes of user interface components that solve the problem of input and output of information for this particular task and creates/modifies the sc-model of the user interface containing all the necessary elements.
- SC-model adaptation agent. It is responsible for adding the properties necessary for interpretation to the user interface components (e.g. size, color of elements, their location, etc.) based on the system settings, knowledge about the environment and rules of adaptation to the environment, knowledge about the user and rules of adaptation to certain user characteristics.
- Interpreter. It converts a specific user interface sc model into an external language compatible with the user interface implementation platform, so that the user interface can be recreated on the platform or the state of an existing interface can be updated.
- User action interpreter. It obtains information about user interactions with the interface and triggers the system to perform the action described in the user