Ontological Methods of Designing User Interfaces

Vladislav L. Litvinov
Saint-Petersburg Electrotechnical University «LETI»
St. Petersburg, Russia
vlad.litvinov61@gmail.com

Abstract— **The development of technologies and the emergence of new requirements for the development of information systems leads to the complexity of interfaces associated not only with the increase in the set of functions of information systems, but also with various changing conditions of their operation. In this paper we consider the concept of extensibility tools for design and implementation of the user interface in the framework of the ontological approach. The analysis of ontological approach to user interface design automation is presented. The methodology of user interface code generation by its project is proposed.**

Keywords— user interface; domain ontology; RDF model; SPARQL

Developing a user interface (UI) is a time-consuming task. According to various experts, on average, it takes at least half the time of software development. To reduce the complexity of the development and maintenance of PI, there are currently various tools for design and implementation automation: wimp interface builders, model-oriented tools and tools based on the ontological approach.

The purpose of this paper is to study the concepts of extensibility of tools for design and implementation of user interface within the ontological approach.

In the framework of ontological approach, interface design is to build its model, i.e. to define all components of the user interface. The interface model is a description created with the help of structural and graphical editors, based on the concept systems of four classes, reflecting the specifics of each component of the interface model. Since the classes of concepts are constantly changing, the extensibility of design tools also implies a significant expansion of these systems of concepts with the corresponding modification of graphic and structural editors controlled by these systems of concepts [1].

To ensure such extensibility of design methods and tools, it is necessary to:

- to present systems of concepts in the form of ontology models;
- provide the interface developer with graphical and structural editors designed to interpret ontology models;
- not to fix rigidly the composition of the models of the ontologies, and to provide the editors of models of the ontology of the subject area to support professional tools.

The PI model is a declarative description that automatically generates the user interface code. It contains only the information that can change when the interface or application requirements change. Each component of the interface model is associated with its own system of concepts, and the component of the interface model is the information presented in this system of concepts (thus, different components of the interface model are defined by different systems of concepts).

There are four main components of the interface model and, accordingly, four classes of concept systems [2].

1. The system of concepts (dictionary of the subject area) of the user, in terms of which he interacts with the application program. In this system of concepts are expressed all input and output data of the application program, as well as information for intelligent support of user actions.
2. A system of concepts in terms of which different types of dialogue are defined (a system of concepts of information representation). This class contains three types of systems of concepts (further the number of such systems of concepts can increase) – a system of concepts of the graphical user interface (based on forms or WIMP-interfaces), a system of concepts of static graphic scenes and a system of concepts for the formation of texts. Thus, each of the concept systems supports the design of its own type of dialogue.
3. A system of concepts for defining a dialogue scenario. It defines the terms used to describe event handlers (the actions performed when events occurred, sources of events, modes of transitions between Windows, methods of selecting instances of the window, etc.).
4. A system of concepts in terms of which the connection between the application and the user interface is carried out. It defines variables, their types of values, and the protocols by which communication is carried out.

The definition of the interface in the framework of the ontological approach assumes the presence of only the information that can change in the life cycle of the information system. Application developers use this knowledge to create appropriate interfaces for presenting data: a reasonable set of data, grouping and sequencing of input elements,

978-1-5386-5824-6/18/$31.00 ©2018 IEEE

displaying/hiding sections, or navigating between pages. This knowledge is implicitly used by the developer and is based on his experience or other rules that are implicit knowledge. The main idea of this approach is to incorporate this semantic knowledge into a data – driven model, along with the processed application data.

An important requirement for the architecture of modern tool complexes is openness. For tools of automatic code generation this is achieved by explicit representation of concept systems in the form of ontologies. The developer is provided with third-party structural and graphical editors to form user interface components controlled by ontologies. This approach allows you to change components without code modification, but if modification of instrumentation requires modifying the code of the interface, the developer can work with the model code generation, which describes the correspondence between the algorithms and business logic and components of a model of the interface. The ontology generation methodology is presented in Fig. 1. For Fig.2, the architecture of the tool complex is presented.

The software package can be developed using the EasyRdf library [3]. EasyRdf is a PHP library designed to make it easier to work with RDF files. After parsing, EasyRdf creates a PHP object graph that can then be used to place the data on the page. There are debugging methods that allow you to check what data is available at design time. Data is usually loaded into the EasyRdf \ Graph object from the original RDF documents downloaded from the Internet via HTTP. The EasyRdf \ GraphStore class simplifies loading and saving data, as well as further work using SPARQL. An example is shown in Fig. 3.

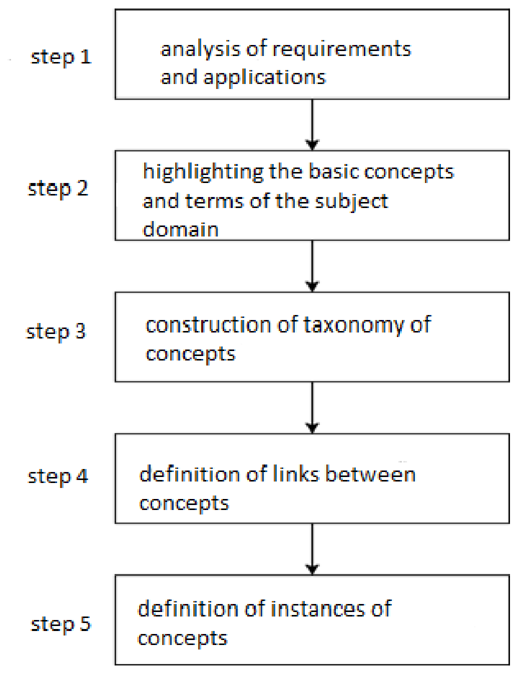

step 1
analysis of requirements and applications
step 2
highlighting the basic concepts and terms of the subject domain
step 3
construction of taxonomy of concepts
step 4
definition of links between concepts
step 5
definition of instances of concepts

Fig. 1. A generalized algorithm for constructing ontologies

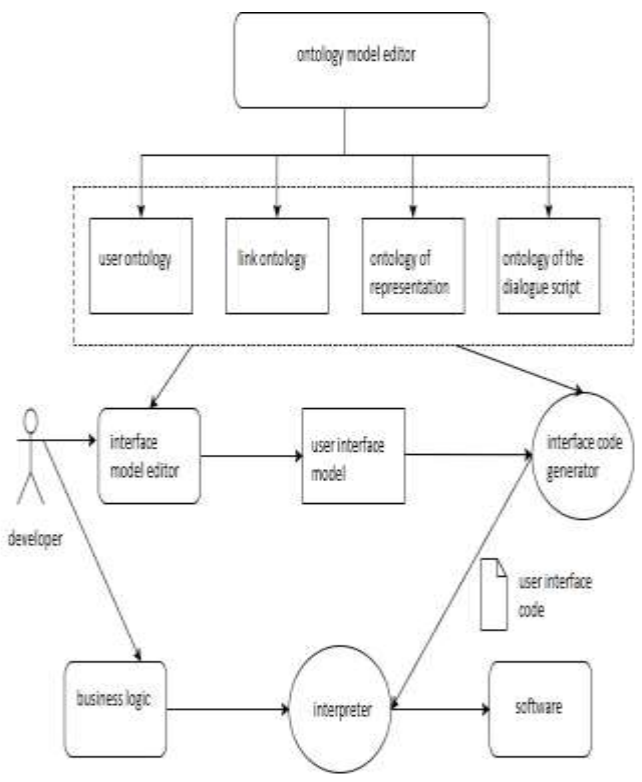
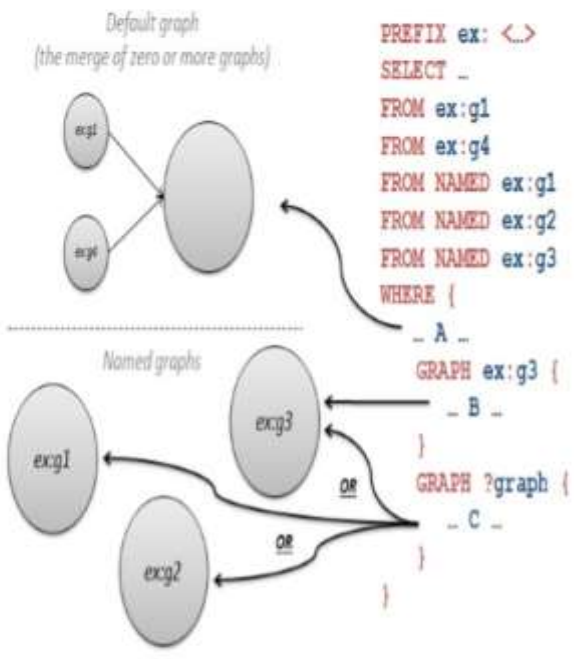


Fig. 2. The architecture of a tool set



```
PREFIX ex: <..>
SELECT ..
FROM ex:g1
FROM ex:g4
FROM NAMED ex:g1
FROM NAMED ex:g2
FROM NAMED ex:g3
WHERE {
  .. A ..
  GRAPH ex:g3 {
    .. B ..
  }
  GRAPH ?graph {
    .. C ..
  }
}
```

Fig. 3. Example SPARQL query to the RDF tree

To describe data elements (i.e. types and constraints of the range or valid values), you need standard and structural information, as well as a meaningful time sequence of screens.

Behavioral information is needed to model dynamic, data-related aspects of the user interface for runtime validation (conditions of existence, activation of elements / groups related to the content of other data elements in the model, indication for complex validation, operations related to data elements and groups caused by changes in input data (reactions) or caused by user actions.

This set of information was considered to be adequate to obtain various aspects of the user interface.

Based on the data obtained, a metamodel was developed, which includes identified information. This model served as the basis for the development of data descriptions for user interface generation (Fig. 4).

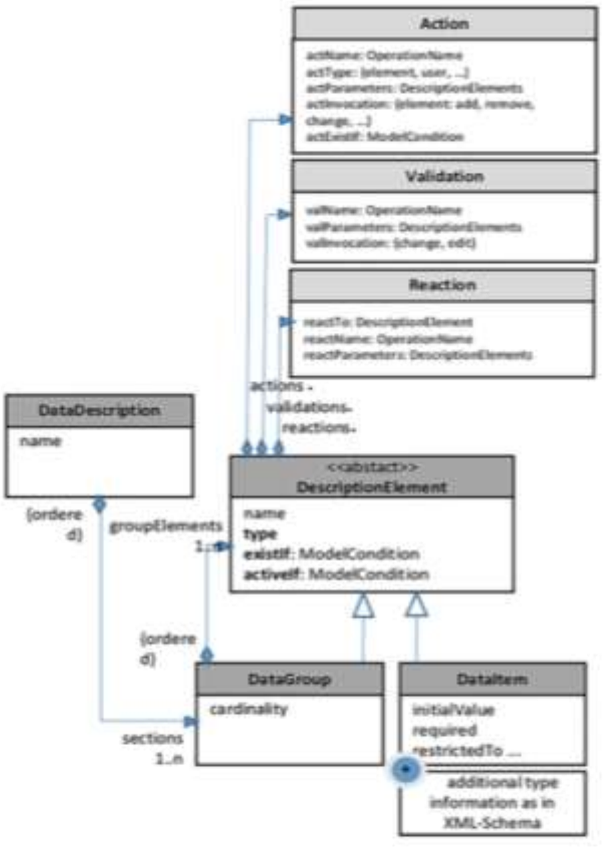


Fig. 4. Metamodel described using UML notation

Thus, the article describes the methodology of automatic generation of the user interface code for its project, presents an analysis of the ontological approach to the automation of user interface design.

References

[1] Gribova V.V., Kleschev A.S. "Control of the design and implementation of the user interface on the basis of ontologies" // Control sciences. 2006, №2, pp. 58-62. (in Russian).

[2] Gribova V.V., Cherkezishvili N. "Automation of development of user interfaces with dynamic data", Open semantic technologies for designing intelligent systems (OSTIS-2011): materials of international. scientific-techn. conference, Minsk, February 10-12, 2011, Minsk: BGUIR, 2011. pp. 287–293. (in Russian).

[3] EasyRdf Documentation. URL: https://github.com/njh/easyrdf.