DIALOG BASED ON GRAPHICAL STATIC SCENES MANAGED BY AN ONTOLOGY

Valeriya Gribova

Abstract. A new method to implementation of dialog based on graphical static scenes using an ontology-based approach to user interface development is proposed. The main idea of the approach is to form necessary to the user interface development and implementation information using ontologies and then based on this high-level specification to generate the user interface.

Keywords: Ontology, interface model, user interface development

ACM Classification Keywords: I.2.2 Artificial intelligence: automatic programming

Introduction

The user interface is an integral part of most software systems. Experts note that complexity and functionality of software systems are increasing every year; at the same time the number of users with a wide range of expertise and, accordingly, requirements to software is rapidly growing. The competition at the software market is increasing, too. All these factors demand a tool capable of realizing dialog between the user and software in accordance with his or her requirements, which are subject to changes during the software life cycle.

Modern tools for user interface development – Interface Builders, User Interface Management Systems, Model-Based Interface Development Environment – are, on the one hand, only oriented to implementation of the GUI (Graphical User Interface) based on using different interface elements – menus, windows, buttons, lists, etc. On the other hand, they do not support design of all user interface components.

To solve the problems mentioned above a new ontology-based approach to user interface development is proposed. The main idea of the approach is to form information necessary for the user interface development and implementation using ontologies and then, based on this information, to generate the user interface. For implementation of different types of dialog (verbal and graphical), ontologies of the graphical user interface and of graphical static scenes on a plane have been developed. They manage design of the presentation component of the user interface and allow us to implement various types of the dialog.

The aim of the present study is to describe implementation methods of various types of the dialog - verbal and graphical - within the limits of the ontology-based approach to user interface development.

The Basic Conception of the Ontology-based Approach

Rapid software progress demands that the cost of interface development need to be decreased, and its maintenance need to be simplified, which is even more important. According to experts, for example, [1] software maintenance exceeds the cost of its development in 3 or 4 times. These requirements in full measure relate to user interface development. The user interface has an additional requirement, namely, adaptability for users with a wide range of expertise.

Taking into account the requirements mentioned above, a new approach to user interface development based on ontologies is suggested [2]. The approach is a modification of the model-based approach to user interface development [3].

The main ideas of the ontology-based approach are as follows: aggregation of uniform information in components of an interface model, formation of information for every component on the basis of the appropriate ontology model and automated the code generation according to this information.

The interface model consists of a domain model, a presentation model, an application program model, a model of a dialog scenario and a relation model.

The domain model determines domain terms, their properties and relations between them. In this system of concepts, output and input data of the application program and information on the intellectual support of the user are expressed.

The presentation model determines a visual component of the interface. It provides support for various types of the dialog.

The application program model determines variables, types of their values shared by the interface and the application program, protocols for communication between the application program and the interface, addresses of servers and methods of message transfer.

The model of a dialog scenario determines abstract terms used to describe the response to events (sets of actions, executed when an event is occurs, sources of events, modes of transfer between windows, methods of the window sample selection and so on).

The relation model determines relations between components of the interface model.

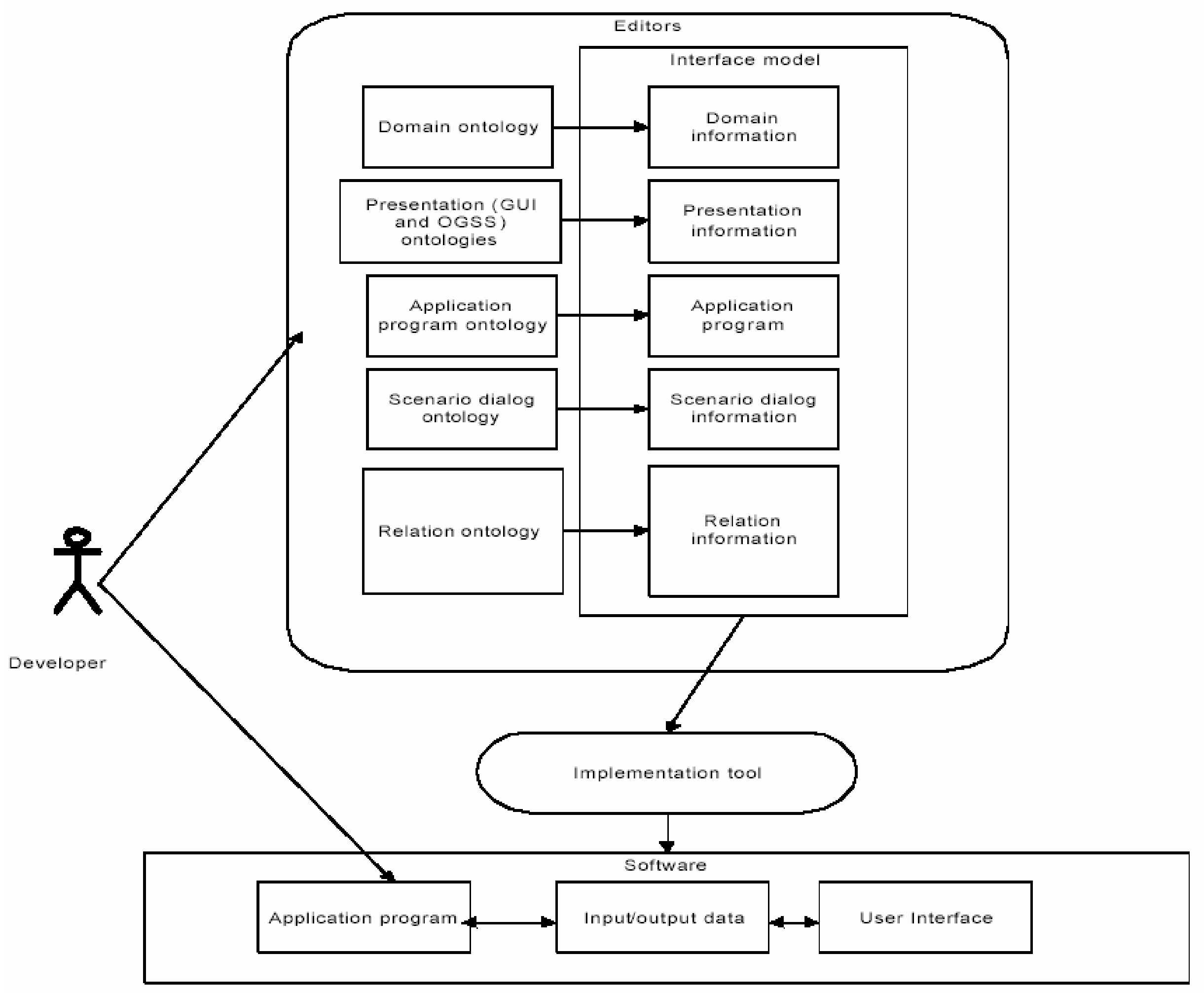


Fig. 1. The basic architecture of the user interface development tool based on ontologies

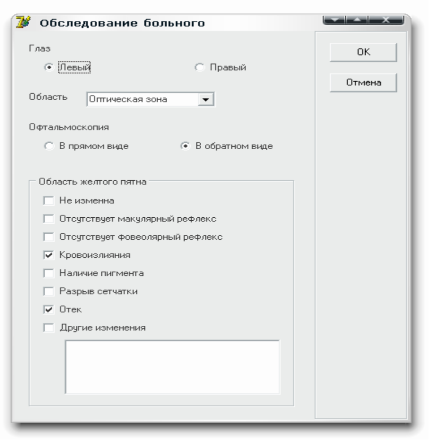
Fig. 1 shows the basic architecture of the user interface development tool based on ontologies. We show only the basic one because (the architecture) as a whole involves additional components such as design critics, advisors, automated design tools, etc. These components are not included in the basic architecture.

An Approach to Implementation of Various Dialog Types

When the user interface is developed, it is necessary that information representing input and output data of an application program be presented in accordance to user requirements and in forms accepted in the domain for which the software is developed.

In this case time various representation forms of information and types of dialog, for example, verbal and graphical, are often required in the framework of the same interface. Fig. 2 shows how the information can be presented to users in various forms. According to our conception, an ontology model for creating and managing every component of the model interface is suggested.

In this way, the interface developer creates domain information. It is a verbal description of domain terms, their properties and relations with other terms. This domain information can be presented in the interface verbally or graphically in various forms depending on user's requirements to representation of information, expertise of users and on their preferences.


Обследование больного

Глаз
[✓ Левый] Правый

Область: Оптическая зона

Офтальмоскопия
В прямом виде [✓ В обратном виде]

Область желтого пятна
- ☐ Не изменна
- ☐ Отсутствует макулярный рефлекс
- ☐ Отсутствует фовеолярный рефлекс
- ☑ Кровоизлияния
- ☐ Наличие пигмента
- ☐ Разрыв сетчатки
- ☑ Отек
- ☐ Другие изменения

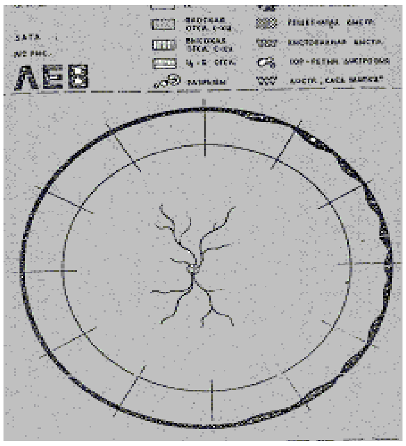


Fig. 2 Representation the same information in various forms

The presentation model is responsible for representation of a domain model. The former is the basis for visual representation of the interface.

To combine various dialog types in the framework of the same interface two ontologies are suggested, the ontology of the graphical user interface (OGUI) and the ontology of graphical static scenes on a plane (OGSS).

The OGUI is intended for presentation of information in the verbal form using interface elements – windows, menus, lists, buttons, etc. The OGUI describes interface elements, their properties and relation to one another. Interface elements permit the user to choose and install values, to start operation and also to move within the program. At present, the OGUI in the OIL language is available in the Internet [4]. The process of design information in the verbal form consists in correlating domain fragments with interface elements and specifying properties of interface elements (dimension, color, location and so on).

The design process in the form of graphical scenes is accomplished by the OGSS. For this purpose terms of the domain are associated with graphical images and their properties peculiar for a particular interface are specified. The generation of graphical scenes, their interpretation (input data for an application program) and automated construction of graphical scenes (output data of a application program are accomplished by the OGSS. Thus, the OGSS is the managing structure for organization of dialog based on graphical scenes.

To provide flexibility and simplify modification of the user interface the correlation between input and output data and term values is established on the basis of the domain model, as the output and input data are independent of the dialog type.

The Ontology of Graphical Static Scenes on a Plane

To implement dialog based on graphical scenes a domain independent the OGSS has been developed. A graphical static scene S on a plane is defined as: S = <B, F, P>, where B is the base graphic representation, F - filler, P - primitive. Fig. 3 shows the OGSS model in the URL language [5].

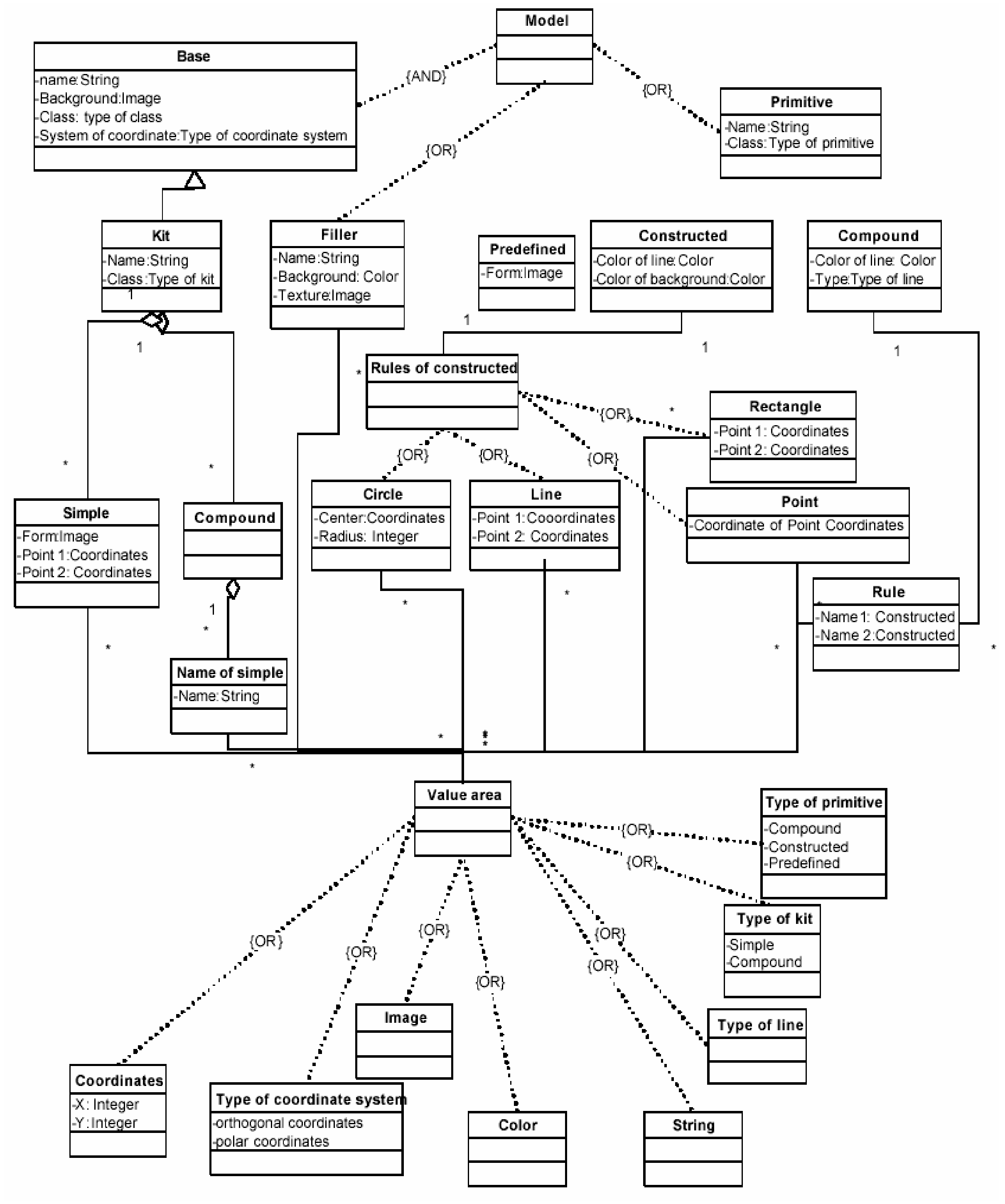


Fig. 3. The model of the ontology of graphical static scenes on a plane

The base graphic representation B, further for brevity named the base, is any graphic figure, scheme, sketch, etc., being a basis for drawing various images on it. The base B consists of the following elements: B = (NmB, ImB, Db, Db′), where NmB is the base name, ImB is the base image, Db is the description of the main elements of the base, and Db′ is the alternative description of the base elements.

The description of the main elements of the base is Db={(b1,nb1), (b2,nb2),…,(bn,nbn)}. Here b1,…,bn are simple elements of the base image, such as $B=b1\cup b2\cup\ldots\cup bn$, i.e. merging simple elements forms the base image; nb1,…,nbn - names of simple elements. The alternative description of elements of the base is Db′={(b′1,nb′1), (b′2,nb′2),…,(b′f,nb′f)}. Here b′1,b′2,…,b′f are compound elements of the base image, nb′1,…,nb′f are names of compound elements of the base image. Each compound element is some merging of simple elements of the base image, i.e.

$b'1=bi1 \cup bi2\cup\ldots\cup bik$, where $1\le k\le n$, bi1,bi2,…,bik, are simple elements of the base image;

$b'2=bj1 \cup bj2\cup\ldots\cup bjh$, where $1\le h\le n$, bj1,bj2,…,bjk are simple elements of the base image;

$b'f=bv1 \cup bv2\cup\ldots\cup bvd$, where $1\le d\le n$, bv1,bv2,…,bvk are simple elements of the base image.

The filler determines possible color and texture options for elements of the base image. A set of possible fillers can be defined for the base F={f1,f2,…,fn}. The filler is defined as fi=<Nfi, Col, Tex>, where Nfi is the name of the filler, Col is the color of the filler, Tex is the texture of the filler.

The primitive determines possible images applied on the base image. A set of possible primitives can be defined for the base: P=p1,p2,…,pn. The primitive pi is defined as: pi=<Npi, Tp, Dp(Tp)>, where Npi is the name of the primitive, Tp is the type of the primitive, Dp(Tp) is the description of the primitive. The type of any primitive can be defined as Tr is predefined, Tb is constructed, Tc is compound.

By the predefined primitive is meant a primitive whose image is known. The description Dp(Tr) of the predefined primitive is Dp(Tr)= < Ip>, where Ip is the image of the primitive.

The constructed primitive it defined by a form, a color of a line and a background. Hence it has the following description: Dp(Tb)=< F, R(F), Cl, Cb>, where F is the form of the primitive, R(F) is the rule for construction of the primitive of a specified form, Cl is the color of the line, Cb is the color of the background. The following forms of a primitive are defined: a circle, a point, a line and a rectangle. For each form, it is necessary to define rules of its construction. The rule of a circle construction is defined by the circle center coordinates and radius: R(circle) =<(x,y), r>; the point is defined by the coordinates: R(point) =<(x,y) >; the line is defined by coordinates of two points: R(line) =<(x1,y1), (x2,y2)>; a rectangle is defined by coordinates of two points, its top left and lower right vertexes: R(rectangle) =<(x1,y1), (x2,y2)>.

The compound primitive is a set of constructed primitives, connected by themselves by lines of a certain color and type: Dp(Tc)=< {(Npi, Npj), (Npj Npk),…, (Npn Npv)}, Cl, Lt>. To describe the compound primitive it is necessary to determine a set of pairs of constructed primitive names (Npi, Npj), that must be connected by lines of a certain color Cl and type Lt.

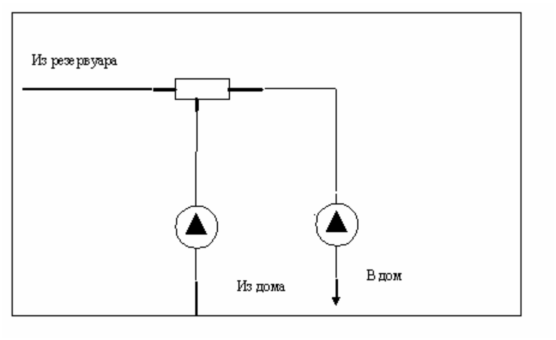
The Design Process of Dialog in the Form of Graphical Static Scenes

The design process of dialog in the form of graphical static scenes is carried out in two phases.

At the first phase, it is necessary for the interface developer to correlate elements the OGSS with the information, specific for a certain domain. In this way, the developer defines a base, i.e. its image, name (a term of the domain), as well as names and images of base components. Further, with the same editor the developer determines fillers and/or primitives by specifying their possible properties.

At the second phase the developer forms the design of the interface, namely, specifies location of the base, primitives, fillers and additional elements of the graphical user interface determined by the OGUI.

The input data dialog of the user with the applied program consists in composing the graphic static scenes. Fig. 4 shows examples of graphical static scenes. According to the specification of the OGSS for a certain domain, the interface recognizes a graphic scene and transfers values of the output data to the applied program in the format assigned by the developer. Then the interface generates graphical scenes based on computation results of the applied program in conformity with the same description.



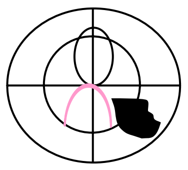


Fig. 4 Examples of graphical static scenes for heat supply and medicine domains.

Discussion

Development of the user interface combining various types of dialog (verbal and graphical) is a topical problem. No less important, as noted above, is the problem of reducing the cost of development and simplifying maintenance of the user interface. In the paper, we have considered how the proposed technology of development allows the specified problems to be solved.

First, the interface is automatically generated based on the declarative description of its model.

Second, information for each component of the model is formed on the basis of the ontology model offered to the developer.

Third, the interface developer can generate various types of dialogues according to requirements of users (verbal and/or graphical on the basis of the OGUI and the OGSS.

Fourth, information transmitted to the applied program and back (the input/output data) does not depend on the form of its representation to the user and is formed based on the domain model. Thus, modification of the dialog does not require any modification of other interface components.

Acknowledgements

The research was supported by the Far Eastern Branch of Russian Academy of Science, the grants № 05-02-01-027, and № 05-01-01-119.

Bibliography

1. Sommerville I. Software engineering. Addison-Wesley Publishing company,1997.
2. Gribova V., Kleshchev A. From an ontology-oriented approach conception to user interface development International //Journal Information theories & applications. 2003. vol. 10, num.1, p. 87-94
3. Da Silva P.P., Griffiths T. and Paton N.W., Generating User Interface Code in a Model-Based User Interface Development Environment, Proc. Advanced Visual Interfaces, V. di Gesu, et al. (eds), ACM Press, 2000.
4. http://interface.es.dvo.ru/ontology.htm
5. The unified modeling language. http://www.uml.org/

Author's Information

Gribova Valeriya – Ph.D. Senior Researcher of the Expert System Department, Institute for Automation & Control Processes, Far Eastern Branch of the Russian Academy of the Sciences: Vladivostok, +7 (4323) 314001