FROM AN ONTOLOGY-ORIENTED APPROACH CONCEPTION TO USER INTERFACE DEVELOPMENT [*]

Kleshchev Alexander, Gribova Valeriya

Abstract: The paper describes a new approach to user interface development which is an evolution of the model-based approach. The aim of the new, ontology-based approach is to eliminate the demerits of demerits of the model-based approach but to conserve its merits and, as a consequence, to lower more the cost of user interface development and maintenance. The main idea of our approach is to exchange models of different interface components for corresponding ontologies. The ontology models accessible by the Internet are used to form the models of there components.

Keywords: Ontology, interface model, user interface development

Introduction

A user interface is a central component of any modern software system. The efforts that are necessary to design, implement, modify and maintain a user interface add up to 70% of all the labour consuming for a software system development. Recent trends in development of software systems generally and of user interface in particular are applying the tools that free developers from low-level programming. These tools based on computer languages of the 4-th generation lower the cost of the development and maintenance for the applied software systems. There are a number of different tools for user interface development. But many user interfaces are implemented with interface builders as before. At the same time, there is a lack of tools that help designers to put all the pieces of an interface design together [1], no support for specifying the dynamic parts, and even for the static parts this support is not adequate [2,3]. These defects have given impetus to the development of a model-based approach for constructing user interface and now several model-based interface development tools have been built, for example [4,5,6,7]. The main goal of this approach is an automatic translation of the declarative, high-level models of interface components into an executable program[8,9]. As a result, the number of procedural components developed in the course of designing an interface becomes considerably less, there is a possibility to reuse the knowledge making up a model, there are powerful tools supporting development [10]. Except these merits, this new technology has also a few demerits that are discussed in [11]. In addition to them it is possible to point out the following. First, up till now there has been no universally accepted standard of interface components. As a result, every model-based tool defines its specific model interface components. In the second place, the methods for implementation of different interface model components by the same model-based tool are different. For every model specific principles and mechanisms are used. This is a reason for difficulties in linking these models together. In the third place, many these tools require the description of application program in detail. This property makes the interface development and maintenance difficult. In the fourth place, different model-oriented tools are based on different declarative languages and data models. This fact makes also a transfer of the same models from one tool to another difficult. In the fifth place, a universally accepted terminology has not been formed within the model-oriented approach yet. As a result, the components, which are identical by meaning and effect, often have different names.

The aim of a new, ontology-based approach to the user interface development advanced in this report is to eliminate the existing demerits of the model-based approach but to conserve its merits and, as a consequence, to lower more the cost of user interface development and maintenance.

THE BASIC IDEAS OF THE ONTOLOGY-BASED USER INTERFACE DEVELOPMENT.

In this report four principal more precise definitions to the model-oriented approach are suggested.

1. A model of a user interface should be considered as a representation of such information about it that should be modified if some conditions of using the software system are changed. The information uniform by meaning should be combined into interface model components.

[*] The presented work has discussed on the KDS-2003. It has corrected in compliance with remarks and requests of participants.

Every component of the interface model should be represented in the form of an ontology model [12,13,14,15]. The representation of knowledge in the form of an ontology model is a universally accepted practice for development of knowledge based systems.

2. The ontology models accessible by the Internet should be used to form the interface model components for which it is possible.

The great current interest in ontologies is caused by the fact that the ontologies of different domains, represented in specific computer languages, can provide access of people as well as computer programs to a huge volume of information and knowledge stored in the Internet and give a possibility to software systems to use these ontologies and knowledge for solving different tasks. It is the ontologies that make possible the development the Internet of the second generation or semantic Internet [16]. The main problem of the semantic Internet is to give direct access of all comers to whole knowledge accumulated by the human civilization.

To implement the approach suggested in this report, it would be necessary to develop, store and maintain the user interface ontology in the Internet. This ontology could be used to form ontologies of particular user interfaces. The user interface ontology as well as the others has to be perpetually maintained.

3. The user interface and application should be designed and implemented as independent components that interact asynchronously through a set of common variables.

This improvement gives a possibility to do away with a task model description and a tool for linking the interface and the application. This idea can permit to decrease the cost of development and to make better interface maintainability.

4. The tool for interface development should be provided with a set of system functions. The interface designer should have a possibility to include necessary system functions in the developed interface according to the customer's requirements.

Any user interface has a set of possible functions according to the functions and tasks of user interface defined, for example, in [17]. These functions are to input data, to solve the task, to exit the application program, to view its results and so on

THE COMPONENTS OF AN INTERFACE MODEL

The user interface model has to contain all the information about this interface that can be modified during the life circle of this interface. This model has also to be appropriate to automatic implementation of the interface (by translation or interpretation).

Before describing components of an interface model we answer the next questions. What is the user interface? What components it consists of? The Xerox Palo Alto Research Center and its official David Liddle have investigated this problem and presented all the user interface components in the form of an iceberg (fig.1). According to the research the interface consists of three main components: a presentation of information to users, an interaction, and a relation among objects. The visible iceberg part is greatly smaller than its invisible part. The iceberg top is information for users (color, animation, objects form, sound, graphic, positioning information). It aggregates only 10% of the whole information and isn’t the main user interface component. The middle user interface part is practice of interaction and feedback with users. It aggregates 30% of the whole information. And, finally, the lower and main iceberg part aggregates 60% of the whole information. It includes objects properties and relations between them. On the basis of the research we determine the components of an interface model. Since a dialog with an software system carried on within the framework of a domain concept system, and the concept system can be modified during the life circle of the software system according to user's wishes and also according to modifications of the domain and of the program functions, the user interface model has to contain some information about this concept system. The domain concept system has to be appropriate to express the input and output data of the system, the information about the application program control, about the interface control and also about an intellectual supporting user's actions.

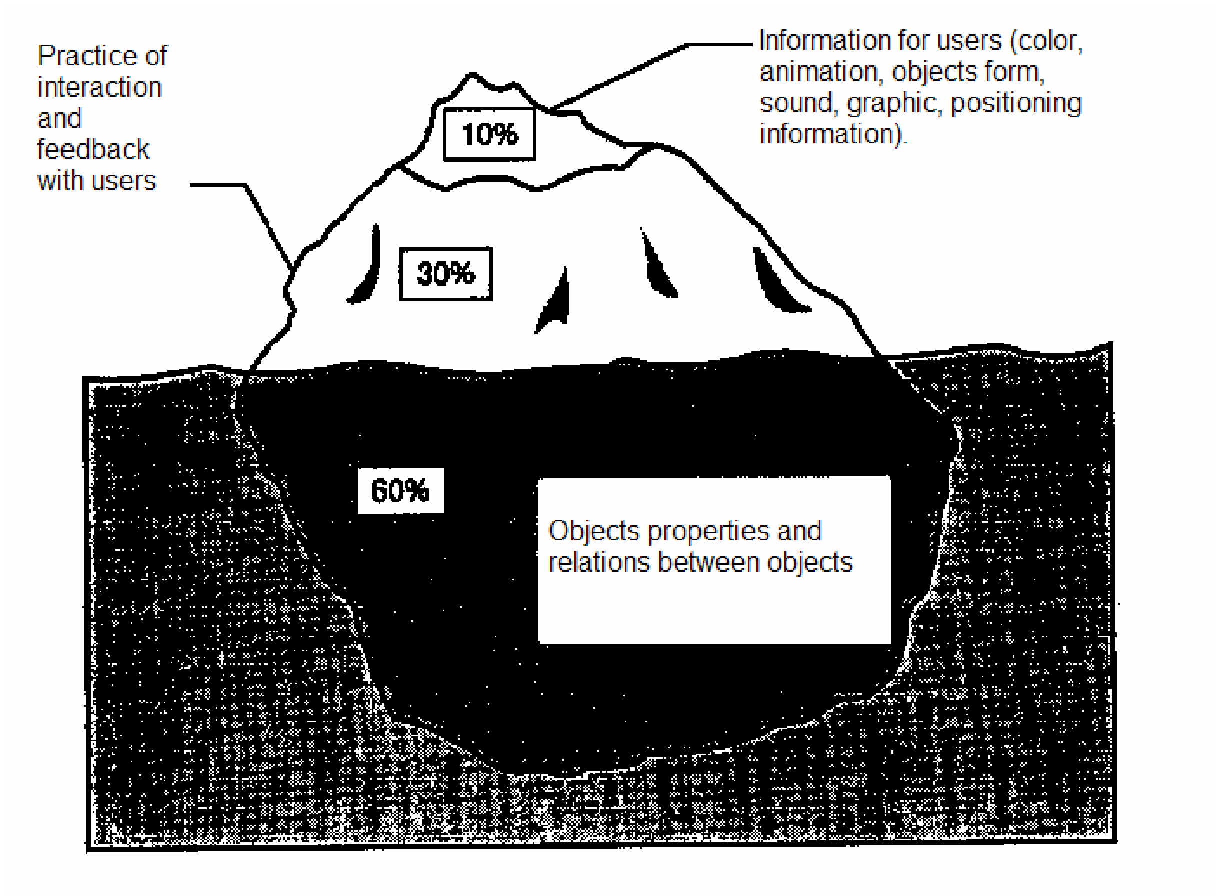
Practice of interaction and feedback with users
Information for users (color, animation, objects form, sound, graphic, positioning information).
10%
30%
60%
Objects properties and relations between objects

Fig. 1. User interface components in the form of an iceberg

Any user's dialog with the system is carried on using some display aspects of the interface such as methods and means for information transmission, for the control of user interaction with the application program, for dialog structures and so on. These display aspects can also be modified during the life circle of the system according to customer's wishes, to modifications of the program functions and to the development of our ideas about the display aspects of the interfaces. So this information about the display aspects has also to be a part of the interface model.

A software system consists of its user interface and application specific program (application program), which the user interface is closely connected with. The application program of the software system can be modified during its life circle according to modifications of the requirements to it. So the information about the application specific program has also to be a part of the user interface model. The less there is this information in the user interface model, the less it is probable that this information will have to be modified when the application specific program is modified, and so the better this part of the user interface model is.

The user interface presents the input and output information of the system to a user in terms of the domain concept system, but to the application specific program in the form of the values of its variables. In this manner there is a correspondence between the domain concept system and the set of the applied program variables. This correspondence can be modified when the domain concept system and/or variables of the application specific program are modified. So the information about this correspondence has also to be a part of the user interface model.

The display aspects of the user interface are used in a dialog to present in a certain form the information that is transmitted from a user to the application program and from it to the user. The user understands this information within the framework of the domain concept system. In this manner there is a correspondence between the domain concept system and the display aspects used in the user interface. This correspondence can be modified according to a modification of the display aspects and of the domain concept system, and also according to user's wishes. So the information about this correspondence has also to be a part of the user interface model.

Any dialog is carried on according to a scenario. This scenario can be modified according to user's wishes, to a modification of the domain concept system and of the application program. So the information about the scenario of the dialog has also to be a part of the user interface model.

The customer's requirements to the set of system functions of the interface can also be modified in the course of using the application program. So the information about the set of system functions of the interface has also to be a part of the user interface model.

In this manner the model of any user interface of an software system can be considered as the set of the following models. They are the models of the domain concept system, of the display aspects for a presentation of these concepts in the interface, of the application program, of a correspondence between the domain concept system and the display aspects, of the scenario of a dialog and of the set of system functions.

The Domain Concept System Model

The direction of attention towards the user means that the interaction between the user and the application program is realized in terms of the domain that the program is intended for. The domain is characterized by its concept system, which consists of concept definitions and of the descriptions for correspondences among them. An explicit representation of these definitions and correspondences is called domain ontology. Thus, a domain concept system model of every user interface is a formal model of a domain ontology.

The concept system used for an interaction between a user and an application program is a part of the concept system of an appropriate domain. It is possible to expect that in the near future formal ontology models for some domains will be presented in the Internet, and the number of models will be increased in time. The terminology of these ontologies will be standardized. If the domain ontology to which an software system relates has been formed, then the ontology of the concept system used by its user interface either is a part of this ontology or can be defined in terms of it.

Domain ontology often has the following property: many its components have the same structure. In this case it is convenient to form its description consisting of two levels. The first one is a reusable metaontology or an ontology of a domain class. It is the same for all the domains of the class. The second one is a domain ontology formed on the basis of the metaontology.

A model for the display aspects of the interface

Now user interface development is an independent branch of software engineering, in which a rather stable concept system has been formed. The display aspects used in every specific user interface can be described as concrete definitions in terms of the concept system. Since designing the display aspects of a user interface is a professional activity, it should naturally be carried on within the framework of this professional concept system.

If this concept system is standardized, reduced to an ontology, formalized and made open to general use, designing the display aspects of the user interface will be considerably simplified. So practical use of the ontology-based approach to user interface development considerably depends on the fact, how quick the open to general use, standardized and formal and ontology model for the display aspects of the user interface will be formed. This ontology model has to contain descriptions for classes of objects having general structure and purpose. It should be possible to expand this ontology by adding new objects and their properties. The content of this branch of knowledge does not depend on a specific interface, but is determined by its achievements. Today a version of the graphic user interface ontology model is accessible by the Internet [18].

In that way, there are two possibilities for designing a model for the display aspects of the user interface in general case. The first one consists in extracting an appropriate concept system from the ontology model for the display aspects of the user interface had presented in the Internet. Extracting this concept system consists in defining the values of all the attributes in the definitions of general concepts. The second possibility is direct forming a specific model using the terminology accepted in this branch of knowledge.

An Application Program Model.

Any software system can be considered as consisting of two main components which are the application program and the user interface. The application program solves some tasks, and the user interface supports an interaction between a user and the application program. According to [17], the user interface is intended for supporting an interaction between a user and an application program, that is a process fulfilling a task. The functions of this interaction are transmission of information for a control of running the application program, transmission of the input data from a user to the application program, of the output data from the application program to the user. In this manner, the interface should transmit the input data and maybe some control information to an application program, and the output data to a user.

An application program model is the better the less it contains information. It is obvious that the minimum information about an application program is a description of all the application program variables by which the exchange of information between the user and application program takes place. It may be considered that every variable is either input or output one, and has an identifier and a possible value range, i.e. an application program model can also be represented in the form of ontology.

When an application program ontology is described, it is necessary to choose an appropriate concept system for the description of the variables. This description is possible:

- - in terms of the domain concept system;
- - in terms of the implementation language;
- - in mathematical terms.

In the first case a transition from the application program ontology model to its implementation cannot be monosemantic. This fact will require an additional description of this transition.

In the second case using different implementation languages will require different models of the application program. This fact can considerably worsen their reusability.

In the third case the description of the variables in mathematical terms is monosemantically understood and reusable.

Thus, designing an application program model reduces to a description of its variables common with its interface in the form of ontology.

The Correspondence between a Domain Concept System and an Application Program Model

There is a correspondence between domain concepts and application program variables. The input data should be transmitted from a user to the application program without information loss, as well as the output data from the application program to the user. In this way, this correspondence between the models of the domain concept system and of the application program has to be defined when an interface model is designed.

The correspondence between a domain concept system model and a display aspect model of an interface

A specific ontology for the display aspects of an interface is a subset of the domain-independent ontology. It determines what interface elements are used in the concrete user interface, and what properties these elements have. The meaning or information components of these interface elements are domain terms described in the domain ontology. In other words, the domain term system forming the input and output data of the tasks is presented to a user in the user interface in the form of different interface elements such as lists, menus, tree control, edit field, graphical images and so on. Thus, there is a correspondence between the domain ontology and the specific ontology for the display aspects of the interface.

At the same time, different communications can exist which communicate the same information. They form a class of equivalent communications. Different users often need in the presentation of the same information in different forms. And what is more, flexibility is an occurring everywhere requirement to the modern interface. It means a possibility of adjusting the interface to user's requirements and its adaptability, i.e. self-adjusting to the user.

Usually a concrete user interface consists of a subset of repeated interface elements, having different meaningful components. For example, they may be menus of the same type which have domain terms as their names and elements. In this case, there is no necessity to give a presentation in the interface for every domain term. It is enough to enumerate all the display aspects of the same type used in the interface and the domain term classes corresponding to them. The domain term classes are defined in the metaontology. But if there is no repetition in the interface, it is possible to assign explicitly every domain term its display aspect.

A dialog scenario model

In previous sections the knowledge branches are presented which are necessary for user interface model development. Besides the declarative model descriptions and the correspondences between them, it is necessary to define the functions for control of the interface by the application program and by the user. To do this, the tool for interface development should have a set of system functions determining user interface behavior. It is necessary to define in a dialog scenario model a correspondence between system functions and their presentation in the interface, and also to define attributes of appropriate system functions.

A DESIGN PROCESS AND IMPLEMENTATION TOOLS FOR THE INTERFACE MODEL

In general case, the design process and ontology-based tool architecture does not differ from the model-based ones. These tools also include design critics and advisors, automated design tools and so on for helping to interface developers.

The user interface design process begins with developing the ontologies or with editing them, i.e. with forming specific ontologies using general ontologies in the case, if the latter have been presented in the Internet. To do this, the tools should contain an ontology editor. The ontology editor gives a possibility to read an existing (for example, in the Internet) ontology and, using it, to extract a specific one. Since these ontologies may appear in the Internet in the near future, their inner representation format has to be standardized. This standardization gives the editor a possibility to read any ontology. If there has not been yet an ontology ready for use in the Internet, then it is possible to form this necessary ontology by the editor, too.

After forming the ontologies of the domain, of the display aspects of the interface and of the application program, models of all the correspondences between the different ontologies are defined by a linking editor. At last, a model of the whole user interface is formed by it. As well as in the case of the model-based approach, an implementation tool generates the source code either in a programming language or in the form of a certain format file, which can be read by an existing UIMS (User Interface Manager Systems). The interface model can also be interpreted at runtime.

Conclusion

The ontology-based approach to user interface development presented in this report permits to decrease the cost of the user interface development and maintenance. The first reason of this fact is reusing ontologies and their fragments both newly developed and presented in the Internet. The second one is using a library of system functions. The third one is using minimum linking between a application specific program and an interface. This property also simplifies the maintenance of software systems.

Bibliography

[1] Puerta, A.R., and Maulsby, D. Management of Interface Design Knowledge with MOBI–D. IUI97:International Conference on Intelligent User Interfaces, Orlando, January 1997, pp.249–252

[2] P. Castells, P. Szekely and E. Salcher. Declarative Models of Presentation. International Conference on Intelligent User Interfaces (IUI'97). Orlando (Florida), 1997, pp. 137-144.

[3] Puerta, A. R. Supporting User–Centred Design of Adaptive User Interfaces Via Interface Models. First Annual Workshop On Real–Time Intelligent User Interfaces For Decision Support And Information Visualization, San–Francisco, January, 1998.

[4] P. Szekely, P. Sukaviriya, P. Castells, J. Muthukumarasamy, E. Salcher Declarative interface models for user interface construction tools: the Mastermind approach.. In Engineering for Humand-Computer

Interaction, L. Bass and C. Unger Eds. Chapman & Hall, 1996 http://www.isi.edu/isd/Mastermind/mastermind-ia.htm

[5] Puerta A. the Mecano project: comprehensive and integrated support for model–based interface development. Computer–aided design of user interfaces, ed. by Jean Vandrdonckt. Pressed Universitaires de Namur, Belgium,1996, pp.19–25.

[6] Lonczewski F., Schreiber The FUSE–system: an Integrated User Interface Design Environment, Proc. CADUI 96, J Vanderdonckt, ed., http://www.info.fundp.ac.be/~jvd/dsvis/cadui96.html

[7] Foley J. History, results, and bibliography of the User Interface Design Environment (UIDE): An Early Model–Based System for user interface design and implementation. Proc. Eurographics Workshop design, specification, verification of interactive systems, F. Patern, ed., 1995, http://www.info.fundp.ac.be/~jvd/dsvis/dsvis94.html

[8] P. Szekely.User Interface Prototyping: Tools and Techniques. 1994 http://www.isi.edu/isd/humanoid-papers.html

[9] 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, 155-160, 2000.

[10] P.Szekely Retrospective and Challenges for Model-Based Interface. 1996 http://citeseer.nj.nec.com/szekely96retrospective.html

[11] Puerta, A.R. Issues in Automatic Generation of User Interfaces in Model-Based Systems. Computer-Aided Design of User Interfaces, ed. by Jean Vanderdonckt. Presses Universitaires de Namur, Namur, Belgium, 1996, pp. 323-325.

[12] N. Guarino, Formal Ontology in Information Systems.Proceedings of FOIS'98, Trento, Italy, 6-8 June 1998. Amsterdam, IOS Press.

[13] Kleshchev A.S. & Artemjeva I.L. Domain ontologies and knowledge processing. Techn. Report, Vladivostok: IACP, FEBRAS, 1999. 25 p.

[14] A. Kleshchev, I. Artemjeva A structure of domain ontologies and their mathematical models. In proceeding of the Pacific Asian Conference on Intelligent systems 2001. Korea Intelligent Information Systems Society. 2001. p. 410-420

[15] M. Uschold Knowledge level modeling: concepts and terminology. The knowledge Engineering Review, Vol.13:1.5-29

[16] www.ontoweb.org

[17] Coats R.B. and Vlaeminke I. Man-computer interfaces. Blackwell Scientific Publications 1987

[18] http;//interface.es.dvo.ru/ontology.htm

Author information

Kleshchev Alexander, Professor, Head of the Expert System Department, Institute for Automation & Control Processes, Far Eastern Branch of the Russian Academy of the Sciences: Vladivostok, +7 4323 310424 kleschev@iacp.dvo.ru, http://www.iacp.dvo.ru/es

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 gribova@iacp.dvo.ru, http://www.iacp.dvo.ru/es