Guarino N. Formal Ontology and Information Systems. In Proceeding of International Conference on Formal Ontology in Information Systems (FOIS'98), N. Guarino (ed.), Trento, Italy, June 6-8, 1998. Amsterdam, IOS Press, pp. 3- 15/

Kleshchev A.S., Moskalenko Ph. M., Chernyakhovskaya M.Yu. Medical diagnostics domain ontology model. Part 1. An informal description and basic terms definitions. In Scientific and Technical Information, Series 2, 2005, №12. PP. 1-7.

Artemjeva I.L., Tsvetnikov V.A. The fragment of the physical chemistry domain ontology and its model. In Investigated in Russia, 2002, 5, pp.454-474. http://zhurnal.ape.relarn.ru/articles/2002/042.pdf

Artemjeva I.L., Visotsky V.A., Restanenko N.V. Domain ontology model for organic chemistry. In Scientific and technical information, 2005, №8, pp. 19-27.

Artemjeva I.L., Restanenko N.V. Modular ontology model for organic chemistry. In Information Science and Control Systems, 2004, №2, pp. 98-108. – ISSN 1814-2400.

Artemjeva I.L., Miroshnichenko N.L. Ontology model for roentgen fluorescent analysis. In Information Science and Control Systems, 2005, №2, pp. 78-88. – ISSN 1814-2400.

Artemjeva I. L., Knyazeva M.A., Kupnevich O.A. Processing of knowledge about optimization of classical optimizing transformations // International Journal on Information Theories and Applications. 2003. Vol. 10, №2. PP.126-131. – ISSN 1310-0513.

Artemjeva I. L., Knyazeva M.A., Kupnevich O.A. A Model of a Domain Ontology for "Optimization of Sequential Computer Programs". The Terms for the Description of the Optimization Object. In Scientific and Technical Information, Series 2, 2002, № 12, pp. 23-28.(see also http://www.iacp.dvo.ru/es/)

Artemjeva I. L., Knyazeva M.A., Kupnevich O.A. A Model of a Domain Ontology for "Optimization of Sequential Computer Programs". Terms for Optimization Process Description. In Scientific and Technical Information, Series 2, 2003, № 1, pp. 22-29. (see also http://www.iacp.dvo.ru/es/)

Kleshchev A.S., Artemjeva I.L. Domain Ontologies and Knowledge Processing. Technical Report 7-99, Vladivostok: Institute for Automation & Control Processes, Far Eastern Branch of the Russian Academy of Sciences, 1999. 25p. (see also http://www.iacp.dvo.ru/es/).

Authors' Information

Alexander S. Kleshchev – kleschev@iacp.dvo.ru

Irene L. Artemjeva – artemeva@iacp.dvo.ru

Institute for Automation & Control Processes, Far Eastern Branch of the Russian Academy of Sciences
5 Radio Street, Vladivostok, Russia

A METHOD OF ESTIMATING USABILITY OF A USER INTERFACE BASED ON ITS MODEL

Valeriya Gribova

Abstract. The article presents a new method to estimating usability of a user interface based on its model. The principal features of the method are: creation of an expandable knowledge base of usability defects, detection defects based on the interface model, within the design phase, and information to the developer not only about existence of defects but also advice on their elimination.

Keywords: Ontology, defects, interface model, user interface development

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

Introduction

Quality and speed of software development are traditionally considered as a compromise where one of them is paid more attention than the other. However, to remain a competitive company developing software should not

only increase speed but also improve quality of its software. To achieve this aim, a lot of efforts of developers are required. According to the Cnews channel in 2001 defects in software cost the world business 175 billion US dollars.

A user interface is an integral part of most software so quality of its development is of critical importance. In addition to general criteria of software quality the user interface has an additional one, namely, usability. The user estimates the whole application program based on its user interface.

Estimating usability is an expensive task in terms of time and labor. This problem is usually solved by increasing the number of testers or by automation of the process.

In this article an additional component of automated detection of usability defects to a tool for user interface development is proposed. The main task of this component is to detect defects of usability in a user interface based on its model and to give advice on their elimination. The paper demonstrates urgency of the problem, the basic idea of the method, and an ontology of defects.

Urgency of the problem

Usability is the measure of the quality of a user's experience when interacting with an application program. It is also a combination of factors that affect the user's experience with the application program, including easiness of learning, efficiency of using, memorability, error frequency and severity, and subjective satisfaction [http://www.usability.gov].

Every year the number of interface elements and their properties is increasing. There are criteria for design of each interface element, their groups and individual characteristics depending on the user's profile (age, experience, specific requirements, etc.), the structure of a domain, a field of using an application program, a type of an application program, and so on. However, all criteria of usability are described in articles, textbooks and manuals informally, as sets of recommendations. The developer must know all these criteria. This fact requires high qualification of developers, their expertise in usability principles, and more evaluators. As a result, cost and time of development increase. To make an application program reliable and to improve its quality, it is suggested to provide the process of user interface development with a system of automated detection of usability defects.

Automation of this process has several potential advantages over non-automated methods, such as [1]:

-Reducing the cost of usability evaluation;
-Increasing consistency of the errors uncovered;
-Predicting time and error costs across an entire design;
-Reducing the need for evaluation expertise among individual evaluators.
-Increasing the coverage of evaluated features.
-Enabling comparisons between alternative designs.
-Incorporating evaluation within the design phase of user interface development.

At present only a few model-based tools for user interface development have facilities for evaluation of a user interface. However, all of them are built into a tool for development and cannot be expanded. These tools quickly become out of day because interface elements are modified, requirements to their design are changed, and new standards are established. So an expandable system of automated detection of usability defects is a problem of urgency.

The Basic Idea of the Method

The principal requirements to a system of automated detection of usability defects are expandability of the system, informing the developer about defects, and giving advice on its elimination.

The author has described a conception of user interface development based on ontologies in [2]. The main idea of this conception is to form an interface model using universal ontology models which describe features of every

component of the model and then, based on this high-level specification, generate a code of the user interface. Components of the interface model are a domain model, a presentation model, a model of linking to an application program and a model of a dialog scenario. Every component of the interface model is formed by a structural or graphical editor managed by a domain-independent ontology model.

Similarly, a presentation model is formed by a graphical editor managed by a graphical user interface (GUI) ontology model. The GUI ontology model describes knowledge required for designing WIMP (windows, icons, menus, and pointing devices) interfaces. It consists of two basic groups of elements (windows and widgets) and three additional groups (control panels, menus and extra elements). Windows are main elements in a user interface since they make up its structure. Other elements are constituents of windows. Widgets (push and radio buttons, checkboxes, lists, etc.) manage an application program and specify properties of objects. Control panels are used to get quick access to commands.

Thus, the GUI ontology model describes interface elements, their properties and interconnections. It is platform-independent and expandable.

Example 1 shows a fragment of the GUI ontology for a text element of a menu.

Example 1. A fragment of the GUI ontology

The example shows the hierarchy of menu elements (see Fig. 1) and description of a text menu element.

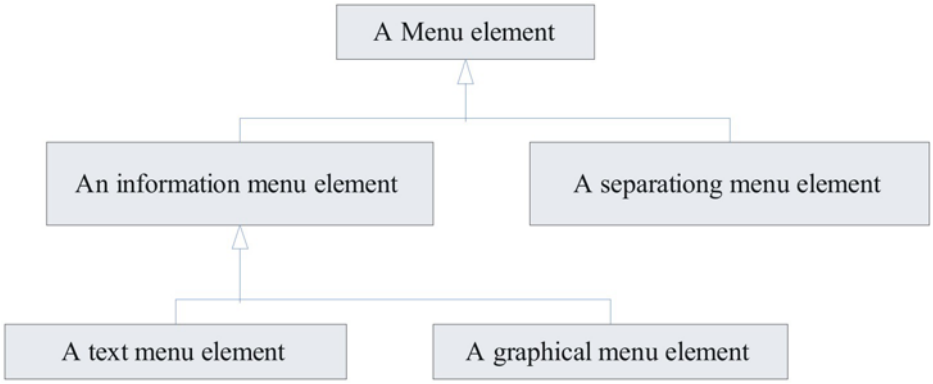

Fig. 1 The hierarchy of menu elements

A text menu element

Description: a class for presenting menu elements with verbal information.

Superclass: an information element of a menu.

Parameters:

Text: describes name to a menu element [type: String]

Prefix: describes prefix of the element [type: String]

Postfix: describes postfix of the element [type: String]

Font: describes font of the element [type: Font parameters]

Background: describes background color of the element [type: color]

A particular presentation component of a user interface model is a subset of the GUI ontology model. It means that to form a presentation component of the user interface model the developer is to determine values of properties of the GUI ontology model. This process requires that the developer should have expertise in usability principles; otherwise a presentation component a user interface model would have defects.

To detect these defects a knowledge base of interface defects has been made. Every element of this knowledge base is linked to elements of the GUI ontology model. It should be noted that since the GUI ontology model is

expandable, when a new element is added to this ontology model, description of a defect in the knowledge base could be modified or a new description of a defect could be added.

There can be two ways to detect defects in the interface model. The first one is detecting a defect in designing an interface element, e.g., when the developer forms a string (a component of an interface element). If the length of this string exceeds a maximal length, the developer can be informed immediately. The second one is checking a set of properties in different interface elements. It is possible only after a fragment of an interface has been designed. For example, to detect a defect of an interface element arrangement in a window it is necessary to design this window first and then to check it. Therefore, the system of automated detection of usability defects is to work in two modes. Fig. 2 shows the basic architecture of the system.

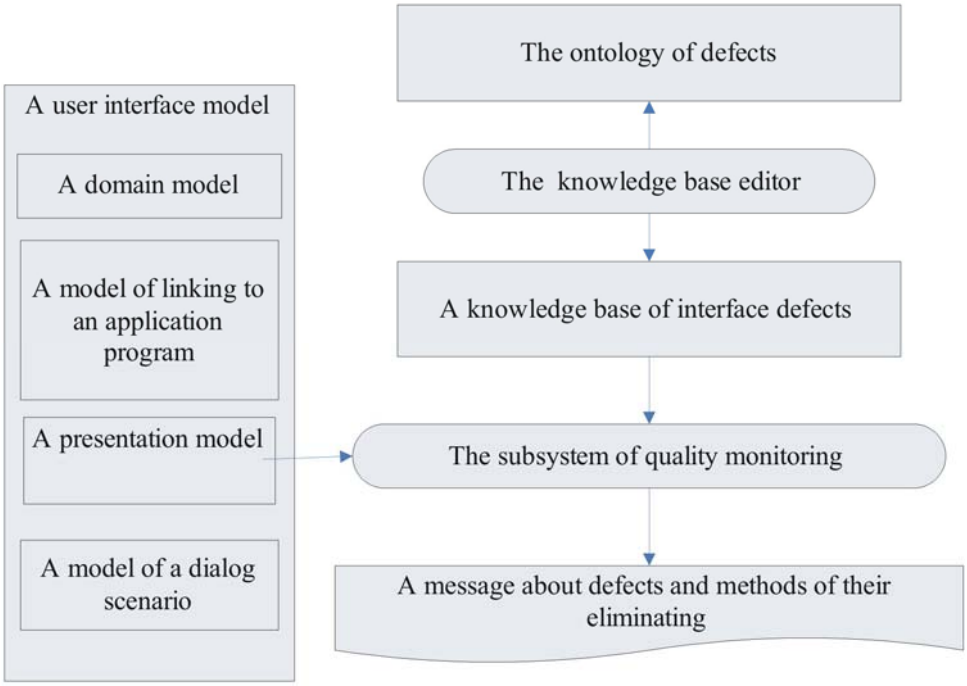

Fig. 2 The basic architecture of the system of automated detection of usability defects.

Ontology of Defects

A defect (fault) is detected in software when the developer makes a mistake due to a typo, poor understanding of some processes, principles, and so on. A defect is a coded mistake of the developer. To detect defects in software it is necessary to accurately classify them. The following ontology of defects is proposed.

1. *Name of a defect*.

2 *Type of a defect*. There may be two defect types, namely, presentation element defects or composition defects. The former occurs in designing an interface element; the latter is found after designing a set of different interface elements.

3. *Name of a class*. It is a metaterm of the GUI ontology model. It indicates a class name of the interface element whose defect is described. This item can contain some classes. On the one hand, an interface element can consist of some classes; on the other hand, when a composition defect is described we must include all classes involved in detecting a defect.

4. *Superclass*. It is also a metaterm of the GUI ontology model indicating a name of a parent class.

5. *Parameters*. There are the parameters that are used for detecting a defect. They correspond to parameters of a class from the GUI ontology model.

6. *Method of detecting*. It is an algorithm of detecting a defect.

7. *Advice*. It is a message to the developer on eliminating a defect.

To illustrate the above, let's consider the following descriptions of defects from the knowledge base based on the ontology of defects.

Name of the defect: too many menu elements.

Type of the defect: a presentation element defect.

Name of the class: a top-level menu.

Superclass: menus.

Parameters: the number of menu elements.

Method of detecting: the number of menu elements>9

Advice. This menu consists of more than 9 elements. It will be difficult for the user to perceive. The number of menu elements should be decreased.

Name of the defect: the window has no name.

Type of the defect: a presentation element defect.

Name of the class: a window.

Superclass: an element of the GUI ontology model.

Parameters: the name (type: Boolean).

Method of detecting: the name=0

Advice. The window has no name.

Summary

In this article an approach to automated detection of usability defects is proposed. The basic idea of the approach is to add a system of automated detection of usability defects to the tool for user interface development operated by a knowledge base of interface defects. The main task of the system is to detect defects in a user interface model within the design phase and to give advice to the developer on their elimination.

At present a prototype of the system has been developed at the Intellectual Systems Department of the Institute for Automation and Control Processes, the Far Eastern Branch, the Russian Academy of Science.

The GUI ontology model and a knowledge base of interface defects corresponding with this ontology are used at the Mathematics and Computer Science Institute of the Far Eastern National University within the course "User Interfaces".

Acknowledgements

The research was supported by the Far Eastern Branch of Russian Academy of Science, the grant «An Expandable System for Quality Monitoring».

Bibliography

1. Ivory, M.Y., Hearst, M.A.: State of the Art in Automating Usability Evaluation of User Interfaces. ACM Computing Surveys, 33 (December 2001) 1–47. Accessible at http://webtango.berkeley.edu/papers/ue-survey/ue-survey.pdf .
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

Author's Information

Gribova Valeriya – Ph.D. Senior Researcher of the Intellectual System Department, Institute for Automation & Control Processes, Far Eastern Branch of the Russian Academy of the Sciences: Vladivostok, +7 (4323) 314001; e-mail: gribova@iacp.dvo.ru; http://www.iacp.dvo.ru/es.