Volodymyr OVSYAK [1,2], Oleksandr OVSYAK [3], Mykhaylo KOZELKO [2], Julia PETRUSHKA [2]
[1] KIELCE UNIVERSITY OF TECHNOLOGY, 7 Tysiaclecia Panstwa Polskiego Ave, 25-314, Kielce, Poland
[2] UKRAINIAN ACADEMY OF PRINTING, 19 Pid Holoskom St., 79-020, Lviv, Ukraine
[3] NATIONAL UNIVERSITY OF CULTURE AND ARTS, 5 Shuchevitsha St., 79-020 Lviv, Ukraine

Models of a Computer System with an Abstract Graphic Element

Abstract

A general model of a computer system with an abstract graphic element and a model of formation of an element form have been designed using the means of algebra of algorithms, the system of algorithmic algebras and platforms.

Keywords: computer system model, abstract graphic element, graph, class diagram.

1. Introduction

The current model of constructing the user interfaces of applied computer systems is based on the use a finite set of standard graphic elements. In particular, the following standard graphic elements are of form: a button, a radio button, elements of alternative choices, text marks and a number of other specific graphic elements. The processes of creating the user interfaces of applied computer system are based on the use of two technologies. First of all it is a coding technology, for example, using C# programming language and Windows Forms environment or other paradigm languages of object programming. Currently, this technology is still used but it is somewhat outdated. The more progressive technology is the technology of dragging standard graphic elements to the form of the applied user interface created by this way. This makes possible an automatic generation of the interface description. For example, in Microsoft Visual Studio .NET environment an automatically generated code of the interface marking up is described in XAML language, which belongs to the paradigm of declarative programming.

Depending on the destination of a computer system, including such as mobile, network and desktop, they have different user interfaces. For example, the interface of a desktop system, when installed in the mobile devices, is necessary to complete taking into account the technical data of these portable devices. A similar situation takes place in solving the inverse problem. Besides it is necessary to consider a horizontal and vertical orientation and the execution of updates of the user interface of applied computer systems. In this regard, an actual task is to create methods and tools for the design of the user interfaces of applied computer systems that would ensure their adaptation to different types of projects without performing reprogramming. In the works [1-5] the solution of this problem is based on the introduction and application of an abstract graphic element. In this paper, the models of fragments of a computer system with an abstract graphic element have been designed. To build a model we have used a system of algorithmic algebras by Hushkov [6], algebra by Ovsyak [7, 8] and instrumental tools of Microsoft Visual Studio .NET platform [9 - 12].

2. A general model of a computer system with an abstract graphic element

A computer system of an abstract graphic element [1-5], which is intended for designing the user interfaces that can be adapted to different types of projects (mobile, network, desktop) without reprogramming is a complex system. In this connection, the method of decomposition has been used for the creation of its model.

2.1. A general model as a formula of algebra of algorithms

The model of the system is described by the operation of cyclic paralleling and it is represented by the formula:

$$S = \varnothing i\ P_i,$$

where S – is a system, $\varnothing$ – is a symbol of the operation of cyclic paralleling, i – is a variable of the operation of cyclic paralleling, $i \in Q = \{0, 1, 2, \dots 13\}$, $P_0, P_1, P_2, \dots P_{12}$ – are subsystems, P_0 and P_1 – are a basic and an auxiliary subsystems of designing the interfaces form, P_2 – is the generation of abstract graphic elements, P_3 – is the generation of standard graphic elements, P_4 – is the location and repositioning of abstract and standard graphic elements on the form, P_5 – is scaling of abstract and standard graphic elements, P$_6$ – is the representation of events of graphic elements and setting the methods of events processing; P_7 – is the representation and editing of properties of graphic elements, P_8 – is the conversion of abstract graphic elements into standard graphic elements, P_9 – is the generation of the developed graphic interface to the desktop system, P_{10} – is the generation of the developed graphic interface to the network system; P_{11} – is saving the graphic interface of the network system, P_{12} – is saving the graphic interface of the desktop system.

2.2. A general model as a formula of the system of algorithmic algebras

Let take the model of the computer system with an abstract graphic element. The computer system is expressed by the use the operation of paralleling, which is a part of the system of algorithmic algebras. The model has the form:

$$S = | \dots ||P_0, P_1|, P_2|, \dots P_{12}|.$$

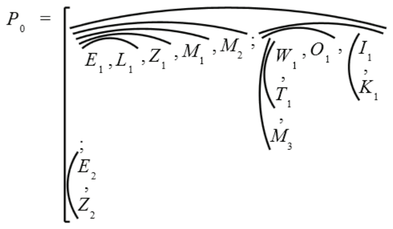
2.3. A general model as a class diagram

If each subsystem is implemented in software by a class, the model of the system as a class diagram will look like in Figure 1. In the diagram the names of the classes MainForm and ParentForm correspond to subsystems P_0 and P_1, Abstract – P_2, Elements – P_3, Move – P_4, Resize – P_5, EventsList – P_6, PropertiesList – P_7, ConvertType – P_8, ConvertToWin – P_9, ConvertToWeb – P_{10}, SaveToWeb – P_{11}, SaveToWin – P_{12}.

3. Models of the subsystem of creating a element window

3.1. A model of algebra of algorithms

The model of computer subsystem of creating a window of the S system is described by the following formula, containing components: E_1 and E_2 – are elements of the interface; Z_1 and Z_2 – are procedures of events processing on these elements; L_1 – is a mark of element E_1; M_1 and M_2 – are procedures of reading and recording the properties; W_1 – are the properties; T1 –is the data input; M_3 – is context data; O_1 – is the procedure of processing the context data; I_1 – is the procedure of initialization of graphic elements, and K_1 – is a root procedure.



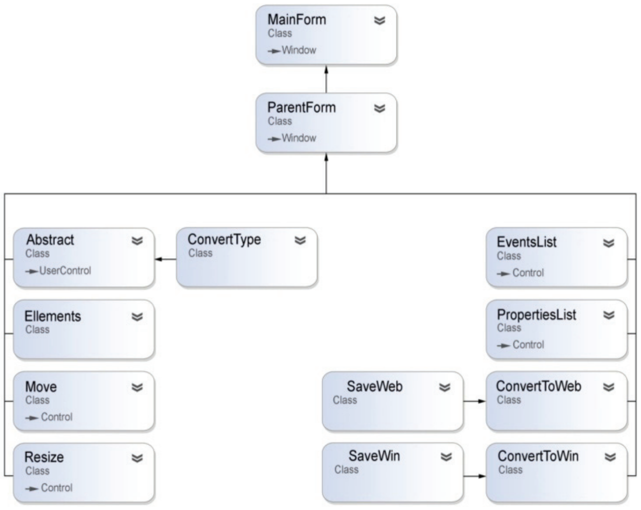


Fig. 1. The model of the system as a class diagram

3.2. A model of the system of algorithmic algebras

To describe the model, we will use the operation of composition (*) and paralleling (||)

$$|E_1 * L_1 * Z_1 * M_1 * M_2 * W_1 * T_1 * M_3 * O_1 * I_1 * K_1,\ E_2 * Z_2|.$$

3.3. A model as a graph

The graph of the subsystem components of the process of the forming of the window is presented in Fig. 2. It contains:

- the procedure of the event processing (button1_Click – Z_1) of the interface element button1 (E_1). The procedure depends on the creating a new window;
- a mark of the element button1 (L_1);
- the procedures get_nazva (M_1) and set_nazva (M_2), which implement the property nazva (W_1), which sets and displays the names of the created elements of window,
- element E_2 (button2), K_1 (NameRoot), Z_2 (button2_Click), T_1 – InitializeComponent, M_3 – _componentLoaded, O_1 – System.Windows.Markup.IComponentConnector.Connect, T_1 – textBoxs1.

4. Models of the generation process of object graphic elements from an abstract element

After setting the type of an object graphic element that must be replaced by an abstract one, we begin the process of generation and replacing an abstract element by an object graphic element.

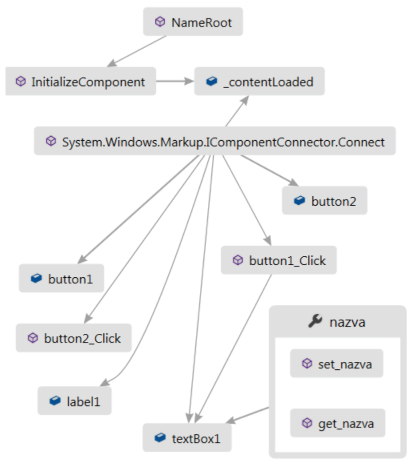
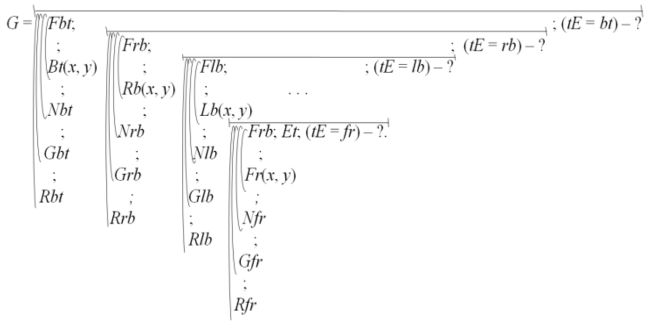


Fig. 2. The graph of the components of the process of a form element design

4.1. A model of algebra of algorithms

The process of generating and replacing an abstract element by an object graphic element is described by model G, which is expressed by the formula:



It includes the recognition of the given type of the object graphic element that must be replaced by an abstract one. The recognition process is described by the operation of elimination with the condition $(tE = j) – ?$. The variable contains the type of the given object graphic element. Possible types of object graphics are run by the variable j ($j \in T = \{bt, rb, lb, \dots, fr\}$, where bt –"button", rb – "radio button", lb – "mark", ... , fr – "form"). If the type of the element is not recognized then Et message is shown. After recognizing the type of a graphic element, its generation begins – $Bt(x, y)$, $Rb(x, y)$, $Lb(x, y)$, ... , $Fr(x, y)$, where x, y – are coordinates of the location of the left upper corner of the element. We set the name of the graphic element – Nbt, Nrb, Nlb, ... , Nfr. The generation function is fulfilled – Gbt, Grb, Glb, ... , Gfr. There happens the placement of the generated graphic element on the root – Rbt, Rrb, Rlb, ... , Rfr.

4.2. A model of the system of algorithmic algebras

The model of algorithmic algebras uses the operations of alternatives and composition and has the form:

$$G = ([tE = bt]\ Fbt * Bt(x, y) * Nbt * Gbt * Rbt, ([tE = rb]\ Frb * Rb(x, y) * Nrb * Grb * Rrb, ([tE = lb]\ Flb * Lb(x, y) * Nlb * Glb * Rlb, \dots ([tE = fr]\ Fr(x, y) * Nfr * Gfr * Rfr, Et) \dots))).$$

4.3. Diagram of sequences

The fragment of the model of replacement of an abstract graphic element by an object one as a diagram of sequences is shown in

Fig. 3. The diagram of sequences contains options for the procedure *childeRoot*..., which corresponds to the model *G*. The recognition of the type of the given element - block *if* and *Catch* – correspond to the operations of elimination of algebra of algorithms and operations of alternatives of the system of algorithmic algebras. The implementation of the generation start is (*AddBut*(*pt.X*,*pt.Y* = *Bt*(*x*, *y*)). During the generation procedure we set the name of the element *ElementName,* corresponding to the model *Nbt.* The direct generation of the given element is performed with the function *CreateItem,* which corresponds to *Gbt* analytical model. Then, the placement of the generated graphic element on the root one takes place.

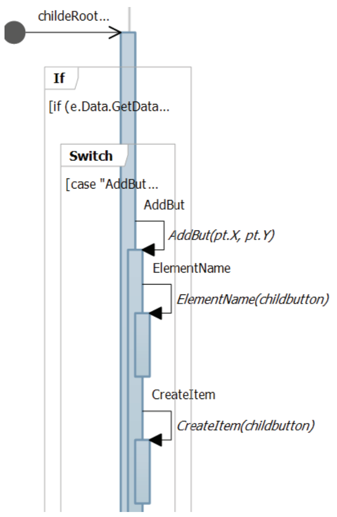


Fig. 3. A fragment of the diagram of sequences of object graphic elements generation from abstract ones

5. Conclusions

The application of algebraic methods for designing models of computer systems, which are algebra of algorithms and the system of algorithmic algebra, gives compact analytical models.

Models of computer systems that are designed using instrumental tools of Microsoft Visual Studio .NET platform are more visible comparing to algebraic models.

The developed models describe the possibility of designing abstract graphic interfaces or their separate fragments or elements on the basis of the introduction and application of an abstract graphic element.

The abstract graphical interface can be converted to interfaces of mobile, network and desktop computer systems or their modifications without altering the software code.

6. References

[1] Kozelko M.A.: Designing of graphic user interface based on abstract elements. Pomiary Automatyka Kontrola, 2013, no 11, pp. 1183 – 1185.

[2] Ovsyak O., Kozelko M., Petrushka J.: Synthesis and optimization of sequencing operation algorithm. Measurement Automation Monitoring, 2015, Vol.61, no 10, pp. 484 – 487.

[3] Ovsyak O., Kozelko M. Petrushka J.: Models of Identification of Sequencing Operation. Measurement Automation Monitoring, 2015, Vol. 61, no 10, pp. 488 – 491.

[4] Kozelko M.A.: The concept of using abstract graphic elements for building GUIs. Computer Science and Information Technologies. Materials of the VIIth International Scientific and Technical Conference CSIT, 2012, Lviv. Publisher Lviv Polytechnic, pp. 60 – 61.

[5] Ovsyak V.K., Kozelko M.A.: Model' abstraktnykh grafichnykh interfeysiv informatsiynykh tekhnolohiy i system. Zbirnyk naukovykh prats' Kompyuterni tekhnolohiyi drukarstwa, 2012, no 28, pp. 99 – 107.

[6] Glushkov V.M., Cejtlin G.E., Jushhenko E.L.: Algebra. Jazyki. Programmirovanie. Izd. Naukova dumka, 1974, pp. 328.

[7] Ovsyak V.K.: Zasoby ekvivalentnykh peretvoren' alhorytmiv informatsiyno-tekhnolohichnykh system Dopovidi Natsionalnoyi Akademii Nauk Ukrainy, 1996, no 9, pp. 83 – 89.

[8] Ovsjak A.V., Ovsyak V.K.: Modificirovannaja algebra algoritmov i instrumentalnye sredstva obrabotki formul algebry algoritmov. Upravljajushhie sistemy i mashiny, 2013, no 1, pp. 27 – 36.

[9] Trojelsen Je.: Jazyk programmirovanija C# 2010 i platforma .NET 4.0. – 2011, Moskva, OOO I.D. Viliams, pp. 1392.

[10] Nejgel K., Iven B., Glinn D., Uotson K., Skinner M., Dzhons A.: C# 2005 dla professionalov. Moskva–Sankt-Peterburg–Kiev, 2007, Dialektika, pp. 1550.

[11] Mak-Donal'd M.: WPF: Windows Presentation Foundation v .NET 3.5 s primerami na C# 2008 dla professionalov. 2008, Moskva: OOO I.D. Viliams, pp. 928.

[12] Nejgel K., Iven B., Glinn D., Uotson K., Skinner M.: C# 5.0 i platforma .NET 4.5 dlja professionalov. 2014, Moskva–Sankt-Peterburg–Kiev: Dialektika, pp. 1435.

Received: 21.07.2016 Paper reviewed Accepted: 01.09.2016

Prof. Volodymyr OVSYAK, PhD, eng.

He specializes in theoretical and applied computer science, theory of algorithms, programming, information systems, mathematical modeling of systems, computer simulation and mathematical modeling. He works as an professor in Kielce University of Technology, Poland.

e-mail: ovsyak@rambler.ru

Mykhaylo KOZELKO, MSc, eng.

He specializes in theoretical and applied computer science, theory of algorithms, programming, information systems, mathematical modeling of systems, computer simulation and mathematical modeling.

e-mail: actionmike@mail.ru

Prof. Oleksandr OVSYAK, PhD, eng.

He specializes in theoretical and applied computer science, theory of algorithms, programming, information systems, mathematical modeling of systems, computer simulation and mathematical modeling. He works as an associate professor in National University of Culture and Arts in Lviv, Ukraine.s.

e-mail: ovsjak@ukr.net

Julia PETRUSHKA, eng.

He specializes in applied computer science, theory of algorithms, programming, information systems, modeling of systems, computer simulation and mathematical modeling. She works as an graduate student in Ukrainian Academy of Printing in Lviv, Ukraine.

e-mail: julja-petrushka@rambler.ru