IMPROVING THE EFFICIENCY OF DESIGNING AUTOMATED INFORMATION SYSTEMS USING INTEGRATED SOFTWARE PACKAGES

Normatova Malika Norqulovna

PhD student in Gulistan State University, Gulistan, Uzbekistan

E-mail: normatova.2989@gmail.com

Abstract. *In this article, researches on increasing the efficiency of automated information systems design processes have been carried out. Methods of optimal use of IDEF0, IDEF3 and DFD methodologies, which represent processes of designing automated information systems, are considered.*

Keywords. *automated information systems, re-engineering, methods, optimization, processes, methodologies.*

Аннотация. *В данной статье проведены исследования по повышению эффективности процессов проектирования автоматизированных информационных систем. Рассмотрены способы оптимального использования методологий IDEF0, IDEF3 и DFD, представляющих собой процессы проектирования автоматизированных информационных систем.*

Ключевые слова. *автоматизированные информационные системы, реинжиниринг, методы, оптимизация, процессы, методологии .*

Annotatsiya. *Ushbu maqolada avtomatlashtirilgan axborot tizimlarini loyihalash jarayonlari samaradorligini oshirish bo'yicha tadqiqotlar olib borildi. Avtomatlashtirilgan axborot tizimlarini loyihalash jarayonlarini ifodalovchi IDEF0, IDEF3 va DFD metodologiyalaridan optimal foydalanish usullari ko'rib chiqiladi.*

Kalit so'zlar. *avtomatlashtirilgan axborot tizimlari, reinjiniring, usullar, optimallashtirish, jarayonlar, metodologiyalar .*

It is not an exaggeration to say that the most difficult design problem is the process of formalizing the subject area. Complex systems, whenever possible, need to be analyzed “from the inside.” A good example here is the process of designing an AIS (automated information system) for a university. Since the analyst can easily put himself in the shoes of any participant in the system, he receives a huge advantage when building a model of the system. Another example is the process of designing a model of particle interaction in nuclear physics, which is complicated by a lack of information about the phenomenon being studied (problems of measurement, experiment, and many others).

But even in the case of complete information about the subject area, the problem of formalization occurs, since in a certain sense there is a barrier (at best, a border) between the scientist (the subject specialist) and the engineer (the one who develops the AIS). In [1] it is emphasized that “the possibility of alienation of professional knowledge from its carriers is determined by the possibility of formalizing this knowledge.”

The general diagram of the modeling process can be presented as in Figure 1, which shows the direction of formalization of the modeling object. Knowledge and experience of the subject area must be transferred from the “carrier” to the engineer who directly creates the AIS.

To carry out this transfer, some help is needed, an intermediate link is needed that connects two opposite principles. The opposite lies in the different approach to modeling: for a scientist, the model is adjusted to reality, and for an engineer, reality is adjusted to the model [3].

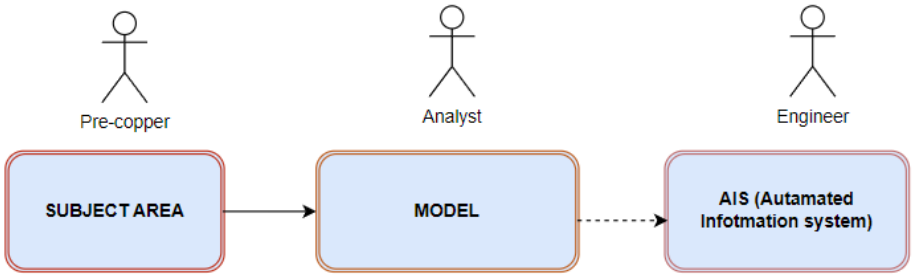

Fig.1. AIS modeling process

The analyst becomes an intermediate link in design. He, together with the subject specialist, develops (describes the existing) structure of the system in the form of interconnected logical units. IDEF0 – IDEF3 – DFD modeling has become a common practice that has gained enormous popularity. Based on the developed model, the engineer and the analyst develop an object model suitable for the development stage. Thus, the analyst must understand both the subject area and implementation issues (for example, the principles of object-oriented programming).

To build a functional model, analysts most often use CASE tools that implement IDEF functional modeling – BPWin and ERWin [3]. The main advantages are clarity of presentation, the possibility of multiple decomposition, openness of the technology. In addition, the specified

Automation tools open up new possibilities for processing model data.

The role of an engineer is often played by an architect or IT department manager. Its task is to identify from the functional model objects of information exchange, information channels and data circulating in the system, as well as systematization and classification of objects of the information system.

Object-oriented modeling in UML is clearer and closer to engineers. There are quite a lot of applications that implement UML modeling, but the most widespread are Rational Rose and Visio. People who directly carry out development - programmers - tend not to use any CASE tools and platforms other than the development environment (for example, Visual Studio). Consequently, the tasks of the architect (manager) include dividing all implemented functionality into small parts and distributing them among project participants. It is known that the more transformations information undergoes, the more likely it is will change. In other words: the more stages of formalization, the less adequate the model is.

In order to reduce the number of information transformations and increase the efficiency of the design process, the following design methodology is proposed, aimed at interfacing with the development stage. The analyst, together with the subject specialist, builds the functional structure of the system in IDEF0 notation. IDEF allows you to decompose functions with the required level of detail, but, at the same time, does not regulate when to stop.

The purpose of this approach is the ability to compare business processes with the functions of the developed AIS in a certain programming language. The meaning of the comparison is as follows. We will consider a programming language function (procedure) as an image of a business function, decomposed according to the described principle. Figure 2 shows a typical match. The function itself corresponds to the business process, the input arcs (or resources) correspond to the

input arguments of the function, and the output arcs of the function block correspond to the output values (in many programming languages, for example in C#, the syntax allows more than one output parameter).

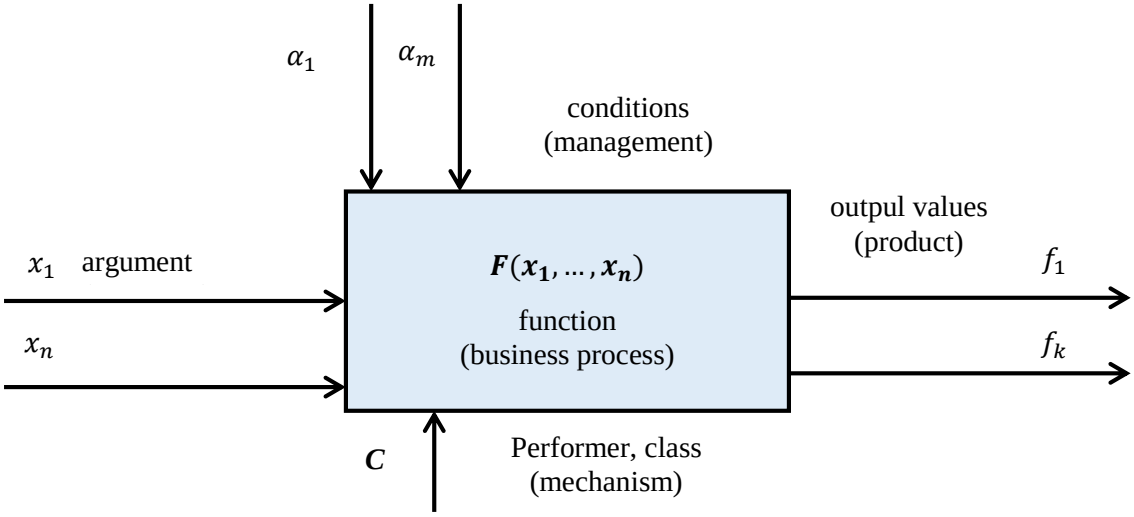

Fig.2. Business process compliance with software function

The correspondence of the arc of the mechanism deserves special attention. We will compare it with the object (programming language class) to which this function should be attributed.

Since a function can belong to only one class, then the mechanism for each decomposed business process must remain strictly one. The resulting decomposition can be quite branched and difficult to understand. To record and process model data, you should use a CASE tool with a developed API (Application Programming Interface) to be able to create custom add-ons. Custom add-ons mean additional functionality that is connected in the form of a plugin or extension library. Additional functionality (in the absence of a standard feature) is required to select (and mark) from the entire set of flows - flows of information, as well as assigning to each mechanism the name of the implementing class. Figure 3 shows a fragment of the diagram in IDEF0 notation, in which information flows are marked with asterisks (*), and the names of the implementing classes to which the corresponding functions are assigned are written in italics. Such additions are allowed because IDEF is an open standard.

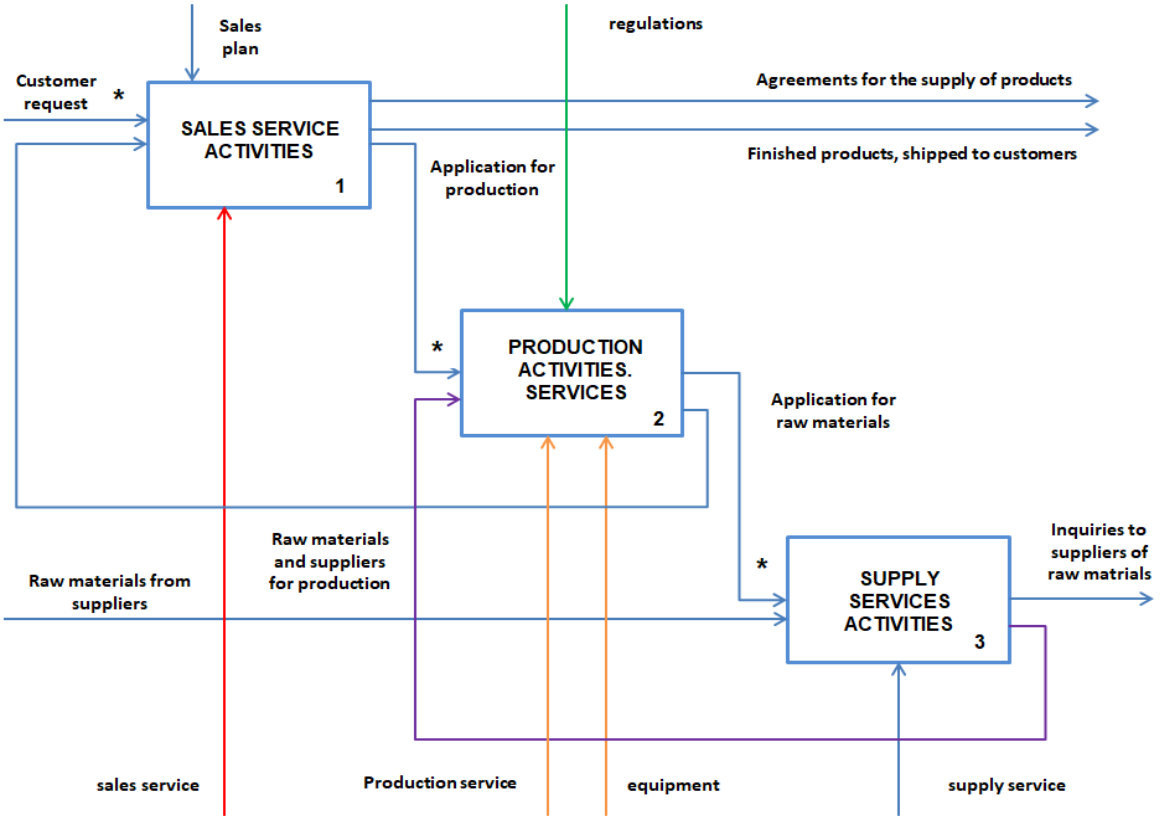

Fig.3. Diagram in extended IDEF0 notation

In Figure 3, the "production service activities" block has two mechanisms. In order to attach a function to a specific class, it is necessary to decompose this block. Upon further decomposition, it may turn out that a certain block is overly detailed, and then it will be necessary to combine the existing blocks together. Multiple transformations of the structure of the designed system are unthinkable without the use of tools. Modern AIS is constantly in motion, i.e. subject to structural and functional changes. This phenomenon is associated with the following facts. Firstly, scientific and technological progress does not stand still, and new information technologies are born, replacing the old ones, having better qualities, and devoid of the shortcomings of the old ones. Secondly, during the testing of pilot projects or the operation of implemented projects, the need for new AIS functions often arises. In addition, some functions may not be in demand. In the latter case, it may turn out that it is necessary to reengineer the system (rebuild with a change in concept) [4]. Any change in the concept or model of the designed system inevitably entails a number of changes in the AIS software components. In practice, this most often means modifying an existing or developing a new AIS software package. Refactoring is intended to clarify the structure of the program code, making the code easier to understand and, therefore, easier to maintain. According to the principles of refactoring, software procedures should not be too long (include too many operations) - such procedures should be divided into many small ones (by analogy with business processes - decomposed). Another rule of refactoring is to avoid too complex interaction between classes: if a certain procedure accesses the fields of another class more often than its own ("envious function"), then the latter should be moved to another class.

A gradual increase in the adequacy of the developed model (due to the possibility of multiple decomposition of one business process) corresponds to the process of "cleaning" the

program code. In essence, the graphical model in the IDEF0 notation corresponds to the AIS program model (code), and the movement in the structure is reflected in both models.

Consequently, to increase the efficiency of AIS development in the context of the existence of the IDEF0 model, it is advisable to automatically generate program code that models the general structure of the system (based on the belonging of functions to certain classes). Thus, in order to more efficiently design an AIS, it is necessary to create an automated software tool that performs the following functions: Construction (export and import into BPWin) of diagrams in IDEF0 notation. Easily switch between context diagrams. Carrying out decomposition of a sheet block. Implementation of combining several blocks.

Clustering of functions - by classes and components. Generating program code for the AIS structure. The current level of development of information technologies and design technologies makes it possible to develop such a tool. All of the above functionality, for example, can be implemented as a plugin for the Visual Studio development environment. Given the eternal relevance of effective design, developers of platform solutions, having implemented such a product, would find great demand for it among analysts and managers involved in design issues.

LIST OF USED REFERENCES

1. Expert modeling, management, planning and evaluation. – M.: “Os-89”, 2004. – 288 p.
2. Askaraliyev O.U., Zaynutdinova M.B. “Development of the structure of the intelligent data processing system (on the example of the Integrated Management System):, *Bulletin of TUIT: Management and Communication Technologies*. 2020 year.
3. Askaraliyev O.U., Zaynutdinova M.B., «Decision support systems in integrated management», I Mejdunarodnaya nauchno-prakticheskaya konferensiya “science, education, innovation: topical issues and modern aspects” Ühingu Teadus juhatus (Tallinn, Estonia). Deceber 2020.
4. Averkin A.N., Gaaze-Rapoport M.G., Pospelov D.A. Explanatory Dictionary of Artificial Intelligence. M .: Radio and communication, 1992.
5. Gavrilova T.A., Gulyakina N.A. Visual methods of working with knowledge: an attempt to review. Iskusstvenny intellekt i prinyatie resheny [Artificial Intelligence and Decision-making]. 2008, no. 1, pp. 15–21 (in Russ.).
6. Yurin A.Yu., Grishchenko M.A. Knowledge base editor for CLIPS. Programmnye produkty i sistemy [Software & Systems]. 2012, no. 4, pp. 8–87 (in Russ.).
7. Unified Modeling Language (UML). Available at: http://www.omg.org/spec/UML/ (accessed June 2, 2014).
8. Brans J.-P. and Mareschal B. The PROMCALC & GALA decision support system for multicriteria decision aid. Decision Support Systems. Vol. 12, No. 4 / 5. P. 297-310.