Multi-Knowledge Electronic Comprehensive Journal For Education And Science Publications (MECSJ)

ISSUE (14), Nov (2018)

www.mescj.com

USAGE OF PROTOTYPING IN SOFTWARE TESTING

1st Khansaa Azeez Obayes Al-Husseini
Babylon Technical Insitute , Al-Furat Al-Awsat Technical University,51015 Babylon,Iraq.
1st Khansaa_aziz@yahoo.com

2nd Ali Hamzah Obaid
Babylon Technical Insitute , Al-Furat Al-Awsat Technical University,51015 Babylon,Iraq.
2nd alimk_iq@yahoo.com

Abstract: Prototyping process is an important part of software development. This article describes usage of prototyping using Question – and – Answer memory and visual prototype diesign to realize Prototyping software development model. It also includes review of different models of software lifecycle with comparison them with Prototyping model.

Key word: Question – and – Answer , Prototype, Software Development, RAD model.

1. Introduction

One of the most important parts of software development is project design. Software project designing as a process of project creation can be divided in two large parts (very conditional): design of the functionality and design of user interface. To design the functionality, tools such as UML and IDEF0 are used, which have already become industry standards for software development. In the design of the graphical user interface there are no established standards, there are separate recommendations, techniques, design features, traditions, operating conditions for software, etc. At the same time, an important, but not always properly performed, part of this process is prototyping, i.e. the creation of a prototype or prototype of a future system.

Prototypes can be different: paper, presentation, imitation, etc., up to exact correspondence to the future program. Most of the modern integrated development environments for software (IDE) allows to create something similar to prototypes, but it is connected with specific knowledge of IDE and programming language. At the same time, design of the user interface of large software project is usually a task of an individual who does not necessarily participate in programming. Therefore, it is useful to have a tool for user interface prototyping adapted for fast creation of quite complex prototypes. As such tools, various software packages were used: MS Visio, Corel Draw, Adobe Photoshop, Inkscape, GIMP. These programs are not specialized tools for prototyping the graphical user interface, but due to the availability of graphical tools they allow creating acceptable prototypes.

Recently, there are tendencies in the use of specialized tools, adapted specifically for creating prototypes of the graphical user interface. And prototypes can be created for all kinds of software: desktop applications, websites, programs for smartphones.

2. Fundamentals of the methodology of designing automated systems based on CASE-technologies

The increasing complexity of modern automated systems and the increasing demands on them determine the use of effective technologies for creating and maintaining automated systems throughout the life cycle. Such technologies, based on methodologies for the preparation of information systems and corresponding integrated tool complexes, as well as those aimed at supporting the full life cycle of an automated system or its main stages, have been called CASE-technologies and CASE-tools [5].

For the successful implementation of the project of an automated system, complete and consistent, functional and information models of the management system should be built. The accumulated experience of designing these models shows that this is a logically complex and time-consuming work that requires high qualification of the specialists participating in it. However, in many cases, the design of an automated system is carried out mainly on an intuitive level using informal methods based on art, practical experience and expert assessments. In addition, in the process of creating and operating automated systems, information needs of users can be changed or refined, which further complicates the development and maintenance of automated systems. From the listed deficiencies, the approaches based on software and hardware of a special class - CASE-tools implementing CASE-technologies for creation and maintenance of automated systems are most free.

The term CASE (Computer Aided Software Engineering) refers to software that supports the creation and maintenance of an automated system, including requirements analysis and formulation, application software and database design, code generation, testing, documentation, quality assurance, configuration management and project management, as well as other processes. CASE-tools together with the system software and hardware form a complete development environment for the automated system.

One of the basic concepts of the methodology of designing an automated system is the concept of the lifecycle of its software [3].

The software lifecycle is a continuous process that begins when a decision is made about the need to create an automated system software and ends when it is completely taken out of service

The structure of the software lifecycle is based on three groups of processes: the main processes of the software lifecycle (acquisition, delivery, development, operation, maintenance); auxiliary processes that support the execution of the main processes (documentation, configuration management, quality assurance, verification, attestation, evaluation, audit, problem solving); organizational processes (project management, project infrastructure creation, definition, evaluation and improvement of the lifecycle itself, training).

The development covers all the work on the creation of software and its components (analysis, design and programming) in accordance with specified requirements, including the design of the project and operational documentation, the preparation of materials necessary to verify the operability and quality of software projects, materials necessary for the organization of training of personnel, and so on [4].

The operation includes work on the implementation of software components (configuration of the database and user workplaces, provision of operational documentation, training of personnel, etc.), localization of problems arising in the course of operation with elimination of the causes of their occurrence, modification of the software within the framework of the established schedule, preparation of proposals for the improvement, development and modernization of the system. Each process is characterized by certain tasks and methods for their solution, the initial data obtained at the previous stage, and the results. The results of the analysis, in particular, are functional models, information models and corresponding diagrams.

The life cycle of software is iterative: the results of the next stage often cause changes in the design solutions developed at earlier stages.

Several software life cycle models are known. The software life cycle model is a structure that defines the sequence of execution and interrelationships of processes, actions and tasks throughout the cycle. The life cycle model depends on the specifics of the automated system and the specific conditions in which the system is created and functioning.

To date, the following two main models of the life cycle have become most widespread: the cascade method and the spiral model [6].

Cascade model is used, as a rule, for the development of homogeneous automated systems, representing a single whole. Its main characteristic is the division of the entire development into stages, and the transition from one stage to the next occurs only after the work is completed on the current one. Each stage ends with the release of a complete set of documentation, sufficient for the development to be continued by another development team. Advantages of using the cascading method are as follows: at each stage a complete set of design documents is formed, which meets the criteria of completeness and consistency; carried out in a logical sequence of stages of work allow you to schedule the completion of all work and the corresponding costs. The cascade approach has proven itself in the construction of automated systems for which, at the very beginning of the development, it is possible to formulate all the requirements accurately and fully in order to give developers the freedom to implement them technically as best as possible. This category includes complex calculation systems, real-time systems, etc. At the same time, this approach has a number of drawbacks caused primarily by the fact that the actual process of creating an automated system never completely fits into such a rigid scheme, there is a constant need for a return to the previous stages of clarifying or revising earlier decisions [4].

the cascade scheme of development of an automated system can be considered as "simulation with an intermediate control". Inter-stage adjustments provide greater reliability of the cascading model, although they increase the entire development period. The main disadvantage of the cascade approach is a significant delay with obtaining the results. The results are coordinated with users only at the points planned after completion of each stage of work, the requirements for the automated system are "frozen" in the form of a technical assignment for the entire time it was created. Thus, users can make comments only after the work on the system has been completely completed. In case of inaccurate presentation of requirements or their changes during a long period of creating an automated system, users receive a system that does not meet their needs. Models (both functional and informational) of an automated object can become obsolete simultaneously with their approval. The spiral model for the development of an automated system is free from these shortcomings, which focuses on the initial stages of the life cycle: analysis and design. At these stages, the feasibility of technical solutions is verified by creating prototypes. Each coil of the spiral corresponds to the creation of a fragment or version of the software, it clarifies the objectives and characteristics of the project, determines its quality and plans work for the next coil of the spiral. Thus, the details of the project are deepened and sequentially specified and, as a result, a reasonable variant is chosen, which is brought to realization. Iteration development reflects the objectively existing spiral cycle of creating an automated system. Incomplete completion of work at each stage allows you to proceed to the next stage, without waiting for the complete completion of work on the current one. With the iterative development method, the missing work can be performed at the next iteration [8]. The main task is to show the users of the automated system as soon as possible an efficient product, thereby activating the process of clarifying and supplementing requirements. The main problem of the spiral cycle is the determination of the moment of transition to the next stage. To solve it, you need to introduce temporary restrictions on each of the stages of the life cycle. The transition is carried out in accordance with the plan, even if not all of the planned work is completed. The plan is compiled on the basis of statistical data obtained in previous projects and the personal experience of developers of automated systems. Within the framework of the spiral life-cycle model, one of the approaches to software development, known as Rapid Application Development (Rapid Application Development) methodology, was widely adopted. This methodology includes three components: a small team of programmers (from 2 to 10 people); A short but carefully worked out production schedule (from 2 to 6 months); a repetitive cycle in which developers as the application begins to take shape, request and implement in the product the requirements obtained through interaction with the customer. The development team should be a group of professionals with expertise in analysis, design, code generation, and software testing using CASE tools that can interact well with end users and transform their offerings into working prototypes. The life cycle of software in accordance with the RAD methodology consists of four phases: analysis and requirements planning; designing; construction; implementation.

In the phase of analysis and requirements planning, users of the automated system determine the functions that it must perform, identify the most priority of them, which need to be worked out in the first place, describe information needs. Formulation of requirements for an automated system is carried out mainly by users under the guidance of development specialists. The scale of the project of the automated system is limited, the time frames for each of the subsequent phases are established. In addition, the very possibility of implementing the project in specified amounts of funding, on available hardware, etc., is determined. The result of this stage should be a list of priority functions of the future automated system, as well as preliminary functional models of the automated system [7].

At the design stage, some users participate in the technical design of the system under the guidance of development specialists. CASE tools are used to quickly get working prototypes of applications. Users, directly interacting with them, refine and supplement the system requirements that were not identified in the previous phase. The system processes are discussed in more detail. The functional model is analyzed and, if necessary, corrected. Each process is considered in detail. If necessary, a partial prototype is created for the elementary process: screen, dialog, report, eliminating ambiguities or ambiguities. The requirements for differentiating access to data are established. At the same phase, the necessary documentation is being determined. After a detailed determination of the composition of the processes, the number of functional elements of the system being developed is estimated and a decision is made to divide the automated system into subsystems that can be implemented by one development team for a time acceptable to RAD projects (60 to 90 days). Using CASE-tools, the project of an automated system is distributed among different teams (the functional model is divided). The result of this stage should be: a general information model of the system; functional models of the system as a whole and subsystems implemented by individual development teams; Precisely defined with the help of CASE-tools interfaces between autonomously developed subsystems; built prototypes of screens, reports, dialogues. All models and prototypes should be obtained with the use of those CASE-means, which will be used later in the construction of the system. This requirement is caused by the fact that in the traditional approach when information about the project is transferred from stage to stage, uncontrolled distortion of the data is quite often. The use of a single data storage environment for the project allows this to be avoided. Unlike conventional approaches that use specific prototyping tools not designed to build real applications, and prototypes are discarded after eliminating ambiguities in the design of an automated system, in the RAD approach, each prototype is passed to the future system. Thus, more complete and useful information is transmitted to the next phase.

At the stage of construction, the quick preparation of the application itself is carried out. At the same time, the developers perform an iterative construction of a real automated control system based on the models obtained in the previous phase, as well as non-functional requirements. The program code is partially generated by CASE-tools automatically. End users in this phase evaluate the results and make adjustments if the system ceases to meet the previously specified requirements during the development process. Testing of the automated system is carried out in the process of development. After the completion of each separate team of developers, this part of the system is gradually integrated with the rest, the full program code is generated, the joint work of this part of the application is tested, and then the whole system is tested. The physical design of the automated system is completed, including: determining the need for data distribution; analysis of the use of data; physical design of the database; Determine the requirements for hardware resources and ways to increase productivity, complete the development of project documentation. The result of this stage is a ready-made automated system that meets all agreed requirements [7].

During the implementation phase of the automated system, users are trained and organizational changes are made. For this stage, it is characteristic that simultaneously with the introduction of the new automated system, work is carried out with the existing management system until the new one is fully implemented. Since the phase of construction is rather short, planning and preparation for implementation must begin in advance, as a rule, at the design stage of the system. The above scheme for developing an automated system is not final. Various options are possible, depending, for example, on the initial conditions in which an automated system is being created: a) a completely new system is being developed; b) an enterprise survey was conducted and a model of its activity exists; c) the enterprise already has an automated system that can be used as an initial prototype or it must be integrated with the newly developed control system.

3. Software lifecycle models

Under the life cycle model of a software product development is understood the structure that determines the sequence of execution and interconnection of processes, actions and tasks performed during the life cycle of the software product development. The following models of the life cycle of software development were most widely spread (Table 1. Brief characteristics of automated system life cycle models): cascade model, or waterfall model; V-shaped model; prototype model; a rapid application development model, or a RAD model; an incremental model; spiral model [6].

Table 1. Brief characteristics of each of the listed models

Name	Characteristics

Cascade model	Straightforward and easy to use. Constant strict control over the progress of work is necessary. Developed software is not available for changes
V-shaped model	Easy to use. Particular importance is given to testing and comparing the results of the testing and design phases
Prototyping model	A «quick» partial implementation of the system is created before the final requirements are formulated. Provides feedback between users and developers in the process of project implementation. The requirements are not complete
RAD model	Project teams are small (3 ... 7 people) and are made up of highly qualified specialists. Reduced development cycle time (up to 3 months) and improved performance. Code reuse and automation of the development process
Multipass model	A fast working system is created. Reduces the possibility of making changes in the development process. It is not possible to go from the current implementation to the new version during the construction of the current partial implementation
Spiral model	Covers the cascade model. Separates the phases into smaller parts. Allows you to flexibly design. Analyzes and manages risks. Users are introduced to the software product at an earlier stage thanks to prototypes

In homogeneous information systems of the 1970s and 1980s, application software products were a single whole. To develop this type of software product, a cascade model or "waterfall" was used.

The cascade model of the software product is similar to the model of an automated control system.

This process is, as a rule, iterative: the results of the next stage often cause changes in the design solutions developed at earlier stages. Thus, there is a constant need to return to previous stages and refine or revise previous decisions. As a result, the actual development process takes a different form [4].

This model (Fig. 1) was developed as a kind of cascade model, in which special attention is paid to the verification and certification of the software product. The model shows that product testing is discussed, designed and planned, starting from the early stages of the development life cycle.

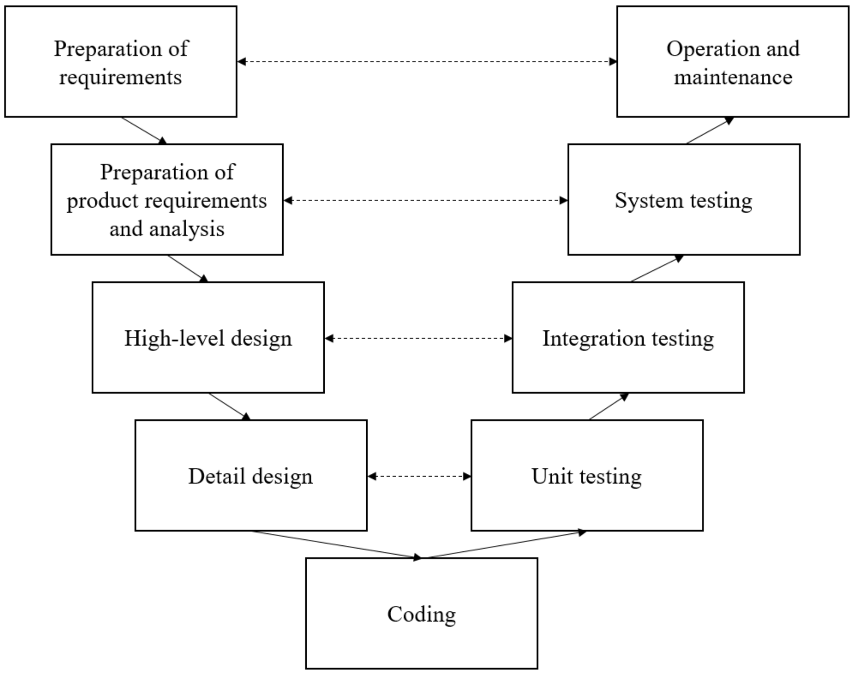
Preparation of requirements
Operation and maintenance
Preparation of product requirements and analysis
System testing
High-level design
Integration testing
Detail design
Unit testing
Coding

Figure 1. V-shaped model

From the cascade model, the V-model inherited a consistent structure, according to which each subsequent phase begins only after the successful completion of the previous phase.

This model is based on a systematic approach to the problem, for which four basic steps are identified: analysis, design, development and review. During the analysis, the project is planned and requirements are drawn up. The design is divided into high-level and detailed (low-level). Development includes coding, review - various types of testing.

On the model, the relationships between analytical phases and design phases that precede coding and testing are well seen. The dashed arrows show that these phases must be considered in parallel.

The model includes the following phases:

Forming of requirements to the project and planning - the system requirements are determined and work planning is carried out;

Forming of requirements to the product and their analysis - a complete specification of the requirements to the software product is compiled [10];

High-level design - determines the structure of the software, the relationship between its main components and the functions they perform;

Detailed design - defines the algorithm for each component;

Coding - converts algorithms into ready-made software;

Unit testing - each component or module of the software product is checked;

Integration testing - integration of the software product and its testing;

System testing - the functioning of the software product is checked after its placement in the hardware environment in accordance with the requirements specification;

Operation and maintenance - the launch of the software product in production. At this phase, the software product may be amended and upgraded.

Advantages of the V-shaped model:

1) A major role is attached to the verification and certification of the software product, from the early stages of its development, all actions are planned;
2) Certification and verification of not only the software product itself, but also all internal and external data obtained are assumed;
3) The progress of the work can be easily monitored, since the completion of each phase is a reference point.

In addition to these advantages, the model has a number of shortcomings:

Iterations between phases are not taken into account; You can not make changes at different stages of the life cycle; testing requirements is too late, so making changes affects the performance schedule [9].

This model should be used in the development of software products, the main requirement for which is high reliability.

4. Prototyping model

The prototyping model allows you to create a prototype of a software product before or during the stage of compiling requirements for the software product. Potential users work with this prototype, identifying its strengths and weaknesses, the results are reported to the developers of the software product. Thus, feedback is provided between users and developers, which is used to change or correct the specification of requirements for the software product. As a result of this work, the product will reflect the real needs of users.

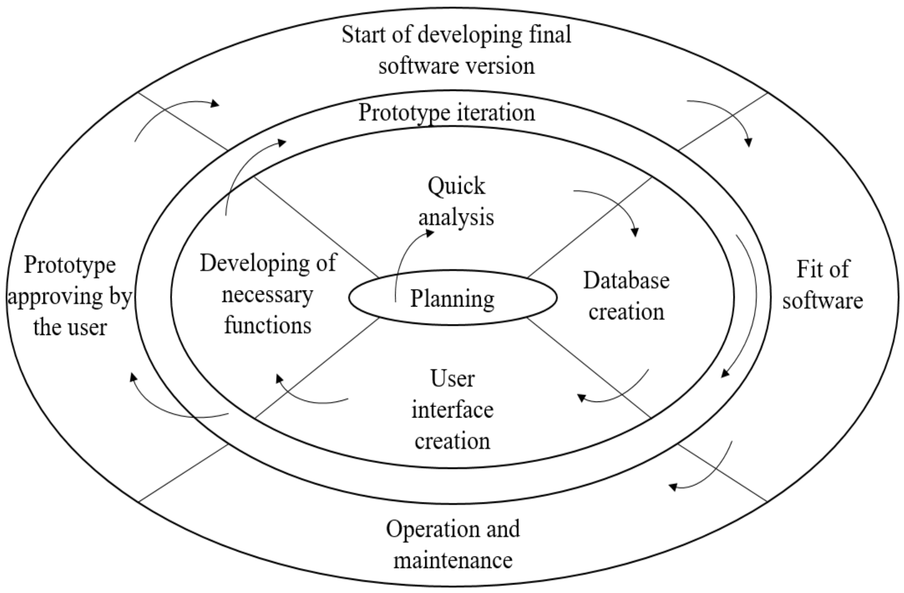
Start of developing final software version
Prototype iteration
Quick analysis
Developing of necessary functions
Planning
Database creation
User interface creation
Prototype approving by the user
Fit of software
Operation and maintenance

Figure 2. Prototyping model

The lifecycle of software development begins with the development of a project plan, then a quick analysis is performed, after which a database, a user interface is created and the necessary functions are developed. As a result of this work, we obtain a document containing a partial specification of the requirements for the software product. This document is the basis for the iteration cycle of rapid prototyping [7].

As a result of prototyping, the developer demonstrates to users a ready-made prototype, and users evaluate its functioning. After this, problems are identified, which users and developers work together to resolve. This process continues until the users are satisfied with the degree of compliance of the software product with the requirements set for it. Then the prototype is demonstrated to users in order to obtain suggestions for its improvement, which are included in consecutive iterations until the working model is satisfactory. After that, users receive the official approval (approval) of the functionality of the prototype and perform its final conversion to the finished software product.

The prototype model has a number of advantages:

1) The interaction of the customer with the system being developed begins at an early stage;
2) Due to the customer's reaction to the prototype, the number of inaccuracies in the requirements is minimized;
3) There is less likelihood of confusion, distortion of information or misunderstandings in determining requirements for software products, which leads to the creation of a better software product;
4) In the course of development it is always possible to take into account new, even unexpected requirements of the customer;

5) The prototype is a formal specification embodied in the software product;
6) The prototype allows very flexible design and development, including several iterations in all phases of the development life cycle;
7) The customer always sees progress in the process of developing a software product;
8) The possibility of contradictions between developers and customers is minimized;
9) Reduces the number of improvements, which reduces the cost of development: emerging problems are solved in the early stages, which drastically reduces the costs of their elimination; customers participate in the development process throughout the life cycle and ultimately are more satisfied with the result of the work [3].

In addition to these advantages of the prototyping model, there are a number of disadvantages:

1) The solution of complex problems can be postponed to the future;
2) The customer may prefer to receive a prototype, rather than a complete full version of the software product;
3) Prototyping can be unnecessarily prolonged;
4) Before starting work, it is not known how many iterations will have to be performed.

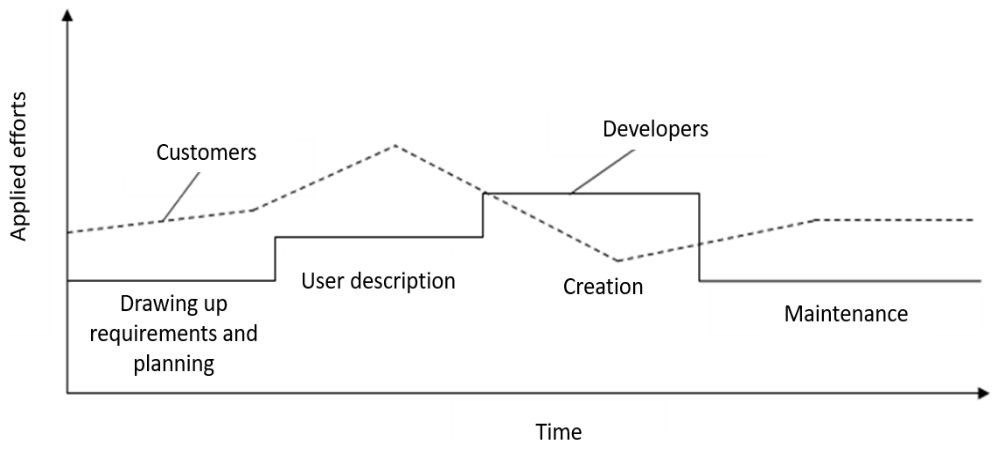

Figure 3. Rapid application development model

The prototyping model is recommended in the following cases:

1) The requirements for the software product are not known in advance;
2) The requirements are not constant or unsuccessfully formulated;
3) Requirements need to be clarified;
4) Need to test the concept;
5) There is a need for a user interface;

6) A new, unparalleled development is in progress;
7) Developers are not sure which solution to choose.

In the RAD model (Fig. 3), the end user plays a decisive role. In close interaction with the developers, he participates in the formation of requirements and approbation of them on working prototypes. Thus, at the beginning of the life cycle, the end user gets most of the work, but as a result, the created system is formed more quickly.

In the traditional life cycle of development, most of the work is done by programming and testing. When automating programming and reusing code used in the RAD-model, most of the work is planning and designing.

In the figure (Fig. 3), explaining the principle of the RAD-model, the stages of the development process are indicated and the participation of customers (a dashed line) is shown on each of them.

The model includes the following phases:

Requirements preparation and planning - are carried out using the so-called method of joint requirements planning (the planning of work on the creation of the software product and the formulation of requirements for the software product are carried out simultaneously), which consists in the structural analysis and discussion of the tasks being solved;

Description of the user - the design of the software product, performed with the direct participation of the customer;

Creation - detailed design, coding and testing of the software product, as well as delivery to the customer;

Maintenance - acceptance tests, software installation and user training.

The model has the following advantages:

1) Using modern tools can reduce the development cycle time;
2) Attraction to the customer's work minimizes the risk that he will remain dissatisfied with the finished software product;
3) The components of existing programs are reused.

At the same time, it has inherent drawbacks [6]:

1) If customers can not constantly participate in the development process, then this can negatively affect the software product;
2) Highly skilled personnel who can use modern tools are needed for work;
3) There is a risk that the work on the software product will never be completed , as it can be looped, so you should always stop in time.

This RAD-model can be used to develop software products that can be modeled well, when the requirements for software products are well known, and the customer can take a direct part in the development process.

5. Conclusion

The life cycle of automated information systems is a continuous process that begins when a decision is made about the need to create an IS and ends at the time of its complete removal from service.

The life cycle model is a structure that determines the sequence of implementation and interconnection of processes, actions and tasks performed during the LC.

The most common are two main models of the LC:

- Cascade model (70-85 years);
- Spiral model (86-90 years).

The structure of the life cycle is based on three groups of processes:

- The main processes of the life cycle (acquisition, supply, development, operation, maintenance);
- Auxiliary processes (documentation, configuration management, quality assurance, attestation, audit, problem solving);
- Organizational processes (project management, project infrastructure creation, improvement of the life cycle, training).

CASE-technology is a technology based on the methodologies for the preparation of information systems and corresponding integrated tool complexes, as well as oriented towards supporting the full life cycle of an automated system or its main stages.

CASE (Computer Aided Software Engineering) means software that supports the creation and maintenance of an automated system, including the analysis and formulation of requirements, the design of application software and databases, code generation, testing, documentation, quality assurance, configuration management and project management, as well as other processes. CASE-tools together with the system software and hardware form a complete environment for the development of AS.

Under the life cycle model of a software product development is understood the structure that determines the sequence of execution and interconnection of processes, actions and tasks performed during the life cycle of the software product development. The following models of the life cycle of the software product development were most widely used: cascade model, or waterfall; V-shaped model; Prototyping model (prototypemodel); model of rapid application development, or RAD-model (rapid application development model); multipass(incremental) model; spiral model.

6. References

- Bashmakov AI, Bashmakov IA Development of computer textbooks and training systems .- M .: *Informatsinno-izdatelskiy dom*.
- Bashmakov IA, Rabinovich PD Analysis of models of semantic networks as a mathematical apparatus of representation of knowledge about educational material // *Handbook. Engineering Journal. No. 7*.- 2002.- Pp. 55-60.
- Buch G., Rambo D., Jacobson A. Language of the UML .- M .: DMK, 2000.- 432 p.
  Vendrov A.M. Designing of software for economic information systems. - Moscow: *Finance and Statistics, 2012*. - 352p.
- Voit NN. Development of methods and tools for adaptive learning of project activities // Information technologies: interuniversity collection of scientific papers .- *Ulyanovsk: UlSTU: 2008*.-Pp. 42-45.
- Gagarina L.G. Fundamentals of technology and software development: *Textbook - M FORUM - INFRA - M*. 2012.
- Dorrer GA, Rudakova GM Modeling of the process of interactive learning on the basis of formalities of colored Petri nets // *Vestnik KrasGU* .- 2004.
- Informatics: A Textbook edited by N. Makarova. - M.: *Finance and Statistics.,* 2011. - 480s.
  Kaner S., Folk D., Ken Nguyen E. Software Testing: Trans. with English. - Kiev: *DiSoft,* 2011. - 544s.
- Semenov M.I. Trubilin IT, Loiko VI Baranovskaya, *TP Architecture of Computer Systems and Networks Study Guide - M Finance and Statistics*, 2013. - 320s.
- Soviets B.Ya. Tsekhankovsky V.V. *Information Technology - M High School*, 2013.
- Sommerville I. Software Engineering. - Moscow: SPB .: Kiev: *Izd. house "Williams*", 2012. - 624s.
- Fridman AL Fundamentals of object-oriented development of software systems. - *Moscow: Finance and Statistics,* 2011. - 200s.
- Sosnin P.I. Pseudo-code control of work flows in the design of automated systems. 2012.
- Al-Husseini, Khansaa Azeez Obayes. Risk Management Tools In The Design of Automated Systems. Interactive Systems: *Problems of Human - Computer Interaction*. – Collection of scientific papers. − Ulyanovsk: USTU, 2017. − 290 p.
- Khansaa Azeez Obayes Al-Husseini ,Information security in the field of technical development and information, Interactive Systems: *Problems Of Human-Computer Interaction Ulyanovsk:* USTU, 71-80 p. 2015.
- Obaid, Ali Hamzah. Information hiding techniques for steganography and digital watermarking. UDC 681.518 (04) Interactive Systems: *Problems of*

Human-Computer Interaction.–Collection of scientific papers.- Ulyanovsk: USTU, 2015, 306 p. 2015.

- Obaid, Ali Hmazah, Tools for conceptual-algorithmic prototyping in solving design problems in the development of systems with software, Interactive Systems: *Problems Of Human-Computer Interaction Ulyanovsk*: USTU, 2017-25-27 P.
- Khansaa Azeez Obayes Al-Husseini / Ali Hamzah Obaid **,** Development of Risk Management Tools in Question-Answering Based Software Design Environment , *International Journal of Computer Science and Mobile Computing* - IJCSMC, Vol. 7, Issue. 6, June 2018, pg.165 – 174 .