COMPUTER SCIENCE

DEVELOPMENT OF SOFTWARE ON THE BASE OF COMPONENT TECHNOLOGIES

*Assistant **Vasila Abasova***
Azerbaijan, Baku, Azerbaijan State Oil and Industry University

DOI: https://doi.org/10.31435/rsglobal_wos/31012020/6887

ARTICLE INFO

Received: 13 November 2019
Accepted: 14 January 2020
Published: 31 January 2020

KEYWORDS

software,
component technology, dispersed systems, Internet.

ABSTRACT

Component technology is currently considered the most advanced approach to software development. The use of ready software components, especially in the construction of large-volume and complex software (such as information systems), is highly effective. The article is of an overview nature and key concepts and principles of component technology, application of component technology in dispersed systems and the Internet, known component-oriented programming technologies are considered in the article.

Citation: Vasila Abasova. (2020) Development of Software on the Base of Component Technologies. *International Academy Journal Web of Scholar.* 1(43). doi: 10.31435/rsglobal_wos/31012020/6887

Introduction. The most time consuming and costly stage of developing all types of information systems (ISs), including corporative information systems (CISs) is software construction. A number of approaches and tools have been proposed to reduce time and cost. The most effective of these are considered RAD methodology [1] and CASE technology, which are aimed at automating the design of the IS [2].

The corporative information system can be developed in two ways: 1) development of the system in stages at the expense of the enterprise's own power (corporation); 2) Obtaining and application of ready CIS. Each of them has its positive and negative features. In any case, however, first, it is necessary to analyze the production activities of the enterprise. Enterprises with the necessary financial means prefer the second method. However, in this case the enterprise must meet the requirements of ready CIS. Ready CIS usually has a module structure, and the application of such a system is carried out in stages, starting with modules to automate more critical areas. In this case, the use of the new features of the modules installed in the corresponding workplaces will ensure the system to be complete. The experience of "ready" IS development allowed to form a new approach to the development of the corporative information system. The basis of this approach is the system's "assembly" of software components produced by different manufacturers. Conctruction of CIS with component technology was made possible by the support of existing software manufacturers in the design and assembly of information systems implemented by various software platforms.

The application area of component technology is not limited to automated systems software. Internet networking capabilities enable the use of component technology in the design of Web applications, such as Internet protocols for communication and HTML pages as user interfaces.

Basic concepts and principles of component technology

A *component* means a separate software block that can be used many times and spreadable part as a compiled code to be used in other programs. A component must have an interface that fully describes its dependence on the external environment to connect to other programs. Interaction of the component with the software environment occurs through *events*. Events are processed in the program using the component.

A set of rules that define the component's interfaces and their implementation, as well as the way for the components to operate and interact with each other in the system, is called a *component model*. Thus, the component model includes the rules that regulate the life cycle of the component, i.e. what situations it goes through when it exists within a particular system (loaded, not loaded, passive, active, cached, etc.).

In addition to the component model, *base services* should be taken into account to ensure inter-component interactions. For example, in a distributed computing environment, components must be able to interact with each other. Such a set of base services and the component model they support is called the *component environment*. Examples of popular component environments include different versions of J2EE, NET, CORBA systems. Relations between components, their interfaces, component model, base services and component environment are shown in Figure 1.

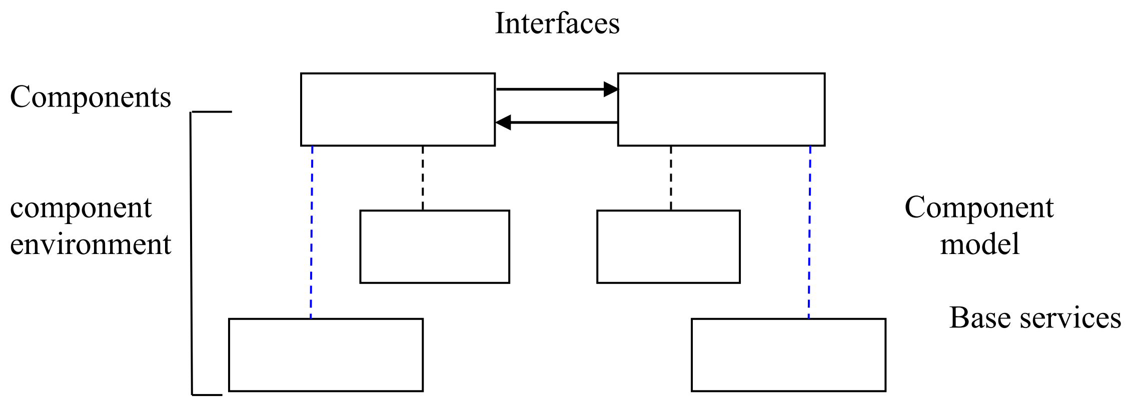

Fig. 1. Basic constituents of component software

Component programming can also be considered as a further development of object-oriented programming (OOP). The key here is to increase the reuse rate of the program code. The component is a larger unit of the program than the object. On the other hand, the component may consist of a large set of classes and, in most cases, does not depend on the programming language.

The core of the component technology is formed by the following principles:

- separation of application logic from general purpose means in general software;
- implementation of application logic in the form of software components;
- organization of inter-component interactions based on standard interfaces.

As a result, it is possible to develop, test and improve functional blocks of the system parallelly and independently of each other.

Application of component technologies in disersed systems

Software of dispersed systems is more sophisticated than other systems and requires high quality. The main problem solved by the dispersed systems is: to provide as many users as possible access to as many resources as possible. The most important features of such systems include transparency, lucidity, scalability and security. *Transparency* is the ability to hide difficulties encountered when working with multiple users at the same time, physical distribution of resources, their duplication, and errors that occur when accessing resources from a user. *Lucidity* assumes the completeness of the interfaces and their description clarity. *Scaling* means changing of the features of the system, depending on the amount of users and resources available, as well as the degree of geographical distribution.

The concept of *security* includes the following characteristics: storage and completeness of data, protection of data and communications, sustainability and ability to recover after errors. These characteristics are due to the multi-user operating mode of the system. Adhering to all of them is a complex issue, and in many cases compromise options are used.

In dispersed systems, the component that initiates inter-component relation is called the *client*, and the component responding, i.e. processing the request is called the *server*. In some cases, the same component can act as both a client and a server. Inter-component relation can be synchronous and asynchronous.

During the *synchronous* relation, the requesting component (client) is blocked and it can resume only after a response is received from the server. An example of synchronous communication is a call to the function or object method in the OOP through a call stack.

After sending a request to the server during *asynchronous* relation, the client can continue working without waiting for the response. An example of an asynchronous relation is an email. Asynchronous relation is more affordable, but it is relatively difficult to use.

In dispersed systems, the *transaction* is used as an information communication unit between components. The basis of the software is formed by the components called *transaction monitors*. They perform queries in the form of transactions on remote access procedures. Such transactions are called dispersed transactions because the processes involved in them can be performed on different machines. A component called "*coordinator*" is used to organize dispersed transactions.

Application of component technologies in WEB software

As mentioned above, the opportunities offered by the WEB service on the Internet contribute to the easy and efficient use of component technologies. Thus, the connection between software components in the Internet environment is created through the Internet's base protocols (TCP/IP, HTTP), and the user interface is provided via HTML that can be viewed by any browser. Tens of thousands of software components can operate simultaneously within the ready framework of the Internet infrastructure to create large-scale software systems, and millions of users can use their services. However, in order to realize the potential benefits of using the Internet, it is important to construct the Web software based on the component technology.

On the other hand, the HTML language gradually shifts its place to extended markup language-XML (eXtensible Markup Language). XML, a universal data format, has a standard lexical form and standard descriptive techniques for describing different structured information. A number of aspects of the development and use of WEB software are related to the way that components exchange information between themselves in different structured data. Using XML allows us to solve some of these problems.

The component systems most commonly used in WEB software are: JavaBeans, COM / DCOM / ActiveX, ROCF (these systems are described below).

Note that in addition to TCP/IP, HTTP protocols, ODBC (Open Data Base Connectivity), JDBC (Java Data Base Connectivity), SAS / SHARE (for working with SAS System data), SAS/CONNECT and other protocols may also be used for create connection between WEB software components.

On the J2EE (Java2Enterprise Edition) platform [3], relation between components running on different processes and machines is carried out in two ways: 1) synchronous relation-by calling methods of Java from a distance; 2) asynchronous relation - through Java data service (Java message service, JMS). *J2EE* is a set of specifications that describe the server platform architecture for enterprise problems in the Java language. Detailed specifications provide scalability of programs and data completeness during system operation. J2EE can be used both on the Internet and on the Intranet. A set of methods concerning to a particular class that can be called from a distance is placed in a special interface called a *remote interface*. The class itself, the methods of which can be called from a distance, must implement this interface. This interface is also implemented through an automated client software.

One of the platforms that support component programming in the WEB environment is *Microsoft NET*. The basis of this platform is the COM (Component Object Model) model, which is accepted as the Microsoft standard of component technology. The COM model is also the base for Active X, OLE and other programming technologies.

One of the important concepts of *Microsoft.NET* is called "*assemble*" [4]. In the component approach, "*assemble*" is a logical unit consisting of a large set of modules required to install the software. The "*assemble*" is characterized by the version IDs and the uniqueness ensured by the digital signature of the author. Based on component technology, "*assemble*" can be used individually or collectively. The description of the assemble is presented in a document called "*manifest*". It stores metadata about the assembly components, author and version IDs, information about types and dependences, usage modes and policies.

Component-oriented programming technologies

Java language based technologies: Being based on dispersed computing technology, Java, a product of the Sun, was accepted by experts and widely spread in a short period of time. Java is often considered not as a programming language, but as a complex technology for creating new generation systems. Specific features of Java-technology are:

- multi-platform: Java-based software can be implemented on different platforms (Unix, Windows, Macintosh, etc.);

- multi-level architecture: system and program architecture can be devided into functional levels - data levels that support existing DBIS; service levels that implement the system and its programs' logic; the image level that implements the user interface;

-multi-component of systems and programs: programs are built via a component approach called "JavaBeans".

JavaBeans is a component technology for multiple use of the Java platform [5]. It is an effective tool for setting dispersed systems that operate in heterogeneous computing environments, both on enterprise and the Internet scale, not depending on devices or operating system. As a result of the further development of this architecture, a new generation, *Interprise JavaBeans (EJB),* of component architecture, was created to be used for enterprise-level problems. JavaBeans and EJB architectures can interact with standard protocols between themselves and other parts of the system. An example of such protocols is the *Internet InterObject Protocol (IIOP)* used in CORBA technology and *JavaRMI (Remote Method Invocation)*, which calls for methods of remote Java objects.

Interaction with non-Java objects is described within CORBA technology and in Java IDL (Interface Definition Language). In traditional SQL-oriented systems, the JDBC protocol can be used to access data.

COM/DCOM/ActiveX technologies: COM is the most widely used component model in the world. COM installs necessary abstracts and rules to define objects and interfaces. It also includes software that performs important functions.

Programs constructed by using COM provide their services through one or more COM objects. Each such object is a copy of a particular class and supports a certain number (at least two) interfaces. Each interface contains one or more methods — a function that can be called by the object's client. In order to call any of these methods, the object client uses an interface index with the same method [6].

Each interface that is supported by the object actually plays a role of contact between the object and its client: the object must support the interface methods according to their purpose, call the client-methods correctly. In order for the contact to work, an agreement must be made between the object and its client to determine the exact identification of each interface, the description of the interface methods, and the specific implementation of the interface.

The name of the interface must be unique. A global unique identifier (qlobally uniqie identifier - GUID) is used for this purpose. GUID having 16-bit value often works with applications, and as it's difficult for users, they give simple names to the interface.

The object and client interface must have a predetermined method of description, that is, the way of determinationt for the methods and their parameters being included into the interface. For this purpose, COM took into account the standard instrument- Interface definition language (IDL). With the help of IDL language, it is possible to create a complete and accurate specification of the COM object interface. In its structure, the IDL language is very similar to C $^{++}$ language. Once the interface is determined, it is impossible to be changed.

In addition to the interface specification, each COM object must support the standard binary format for each interface. Having a standard binary format means that any client, regardless of the programming language, can call any object method.

DCOM (Distributed COM - dispersed COM) provides interaction between components in a network environment. In this respect, DCOM technology can be compared to CORBA technology. Both DCOM and CORBA call the method of an object located on another network machine and provide references to the object.

COM provides transparent access to objects in dynamic libraries of clients and local processes, while DCOM implements transparent access to remote process objects. In this respect, DCOM technology is a partially expanded version of the original COM.

DCOM has "access control services" for security in the creation and access of objects. Despite the complicated moments, DCOM is easy to understand. For those who know the basics of COM, DCOM adds 3 key elements: the way of creating a remote object, a protocol to call that object's methods, and a secure access to it [7].

ActiveX technology is a continuation and development of COM technology, and is considered as the component programming standard. Until 1996, this technology, owned by Microsoft, was called OLE (Object Linking and Embedding).

In terms of a programmer, ActiveX is a black box with properties, methods, and events. ActiveX is fully integrated into Delphi. In terms of the COM objects model, ActiveX's control element is a server that support "Automation" technology and visual editing, being implemented in the form of dynamic libraries and performed in the access filed of an applicaton software.

ActiveX is mainly used by web designers to include multimedia objects into pages [8].

CORBA (Commen Object Request Broker Architecturc) technology: Proposed by the OMG consortium, it plays a role of the technological standard in the construction of dispersed systems. It also combines information technology to meet that standard.

The components of the dispersed systems in CORBA technology are considered as the objects that meet certain interfaces. This technology enables to set those interfaces with the help of an IDL language via the same rule. In this case, the interface and its description do not depend on the programming language used, operating system and computer architecture.

The basis of CORBA technology is the ORB (Object Request Broker), which allows data to be transfered from one object to another. The ORB can be implemented in the form of a software library, as special network service, object-oriented DBIS, or it can be integrated into the operating system. The ORB hides all the details for the implementation of objects, only their interfaces remain out [9].

"CORBA implementation" refers to the existence of a specific ORB and the means of communication with it. Additional utilities are also taken into account, such as a translator that converts an interface written in the IDL language into a program code (Java or C $^{++}$). It is possible to mutually use the objects installed through different ORBs and are available on the network. This opportunity is achieved through a special protocol (GIOP) that implements the interaction between the ORBs. As a result, data and surveys can be transferred from one ORB to another.

Each object is determined by a unique index concerning it. To refer to any object, you need to know its pointer. An object index can be specified by its name or through the interface.

Starting with CORBA 3.0 version, a component model called CCM (CORBA Component Model) was included into it. Its task is to describe the standard framework for the application of CORBA components. The COM model was constructed under the influence of the EJB and is actually an extension not dependent on its language. CCM is an abstract of the essences that provide and receive services (such as reporting, authorization, transaction management, etc.) via precisely defined interfaces (ports).

Conclusions. The development of software based on component technology in large and complicated systems reduces the cost and time required for system installation. Therefore, modern programming technologies are implemented via a component approach.

The essence of component technology is revealed, its application is viewed in dispersed systems and the Internet, and the known component technologies are explored.

REFERENCES

1. Karimov S.G. Information systems. Baku: Elm, 2008.-616 p.
2. Karimov S.G. Management information technologies and corporative information systems. Textbook-Baku, 2010.-426 p.
3. Textbook on J2EE. Overview. http://www.codenet.ru/webmast/java/j2ee.php
4. Component programming in NET. http://www.intuit.ru/department/se/tppobj/17/
5. Component technology of JavaBeans. http://synthess.ipi.ac.ru/sigmod/seminar/s19980326
6. Модель COM/DCOM. http://www.interface.ru/fcet-asp?Url=/borland/com_dcom.htm
7. Component Object Model. http://ru.wikipedia.org/wiki/Component_Object_Model
8. Object Linking and Embedding. http://ru.wikipedia.org/wiki/Object_Linking_and_Embedding
9. CORBA. http://ru.wikipedia.org/wiki/CORBA