International Journal of Emerging Technology and Advanced Engineering
Website: www.ijetae.com (E-ISSN 2250-2459, Scopus Indexed, ISO 9001:2008 Certified Journal, Volume 11, Issue 05, May 2021)
Manuscript Received: 15 April 2021, Received in Revised form: 12 May 2021, Accepted: 17 May 2021 DOI: 10.46338/ijetae0521_02

Applying an Aspect-Oriented Approach When Developing an E-Commerce System

Olga Kucheriava[1], Larisa Bachynska[2], Nataliia Holeho[3]

[1,3]National Aviation University, 1, Liubomyra Huzara ave, Kyiv, Ukraine 03058.
[2]National University of Kyiv-Mohyla Academy, 2 Skovorody vul., Kyiv, Ukraine 04070.

***Abstract*— A new promising approach to program development has been considered – aspect-oriented programming, intended for automated adding to the target applications of crosscutting functionality. The article presents the application of an aspect-oriented approach for creating logging in the e-commerce system.**

***Keywords*— Advice, Aspect, Aspect-Oriented Programming, Crosscutting Concerns, Join Point, Logging, Module, Point Cut.**

I. Introduction

Object-Oriented Programming (OOP) is currently a technology that benefits to most of the new software development projects. Of course, the technology of the OOP has demonstrated its strength in modeling the overall behavior of the system being developed. However, as you can see from the experience, the OOP does not sufficiently handle the growing complexity of software systems.

Aspect-oriented programming (AOP) is one of the concepts of programming which is the further development of procedural and object-oriented programming. This technology allows to reduce time, cost and complexity of development of modern software. Aspects are separate parts that are responsible for one or another program functionality, the implementation of which is dispersed by program code, but consists of similar parts of the code. According to experts, most of the time in projects is spent on maintaining and making changes to the finished program code. Therefore, in the near future, the AOP and other similar transformational approaches plays a very important role. This comparatively new technology has already become quite widespread, showing its effectiveness on test applications, but the place of this approach in the software industry for a number of objective reasons is still not defined.

An essential feature of the software system is the level of difficulty. It is almost impossible for one developer to cover all the details when creating the system, and the complexity inherent in most modern software systems. The complexity of software systems is due to four main reasons:

the complexity of the real subject area, which is the source of the order for development; the complexity of managing the development process; the need to ensure sufficient flexibility of the program; unsatisfactory ways of describing the behavior of large discrete systems [1].

In the course of development, to overcome the complexity of the software system, the proven principle of modular decomposition is used. Modular decomposition involves the division of systems into hierarchical (ie, modular) elements (modules, components, classes, objects, functions, procedures, etc.). "Hierarchy" means that an element can consist of other elements, the latter - even a few elements, etc. The boundaries of these elements are set in such a way that they are related to each other ("neighbors"). In order to provide a close link between elements within each individual element and to minimize the interaction between elements, it is necessary to focus on solving a single problem. Dijkstra called this problem the principle of division of tasks or concepts (principle of separation of concerns) [2]. Being one of the fundamental principles of engineering, the division of tasks (concepts) is widely used at all stages of the development of software systems.

II. Analysis of Studies and Publications

The main ideas of aspect-oriented programming were formulated in the article by G. Kiczales [3]. Useful and up-to-date information on the development and innovation of this programming concept is available on a dedicated website [4], which is constantly updated.

At the heart of aspect-oriented programming are the ideas that occur in the field of programming technology, for example: subject-oriented programming, composite filters (composition filters), adaptive programming (adaptive programming) [5-6]. AOP is closely related to intentional programming, the concepts of which are described in Ch. Simony [7]. Another related concept is the so-called generative, or transformational programming (genera-tive programming, transformational program-ming) [8].

E.A.Zhuravliov and V.A.Kiryanchikov article [9] offers a brief introduction to the problems of the AOP, presents the key concepts of the AOP and discusses ways of integrating aspects, including at the stage of program execution. There [10] are advantages of the aspect-oriented approach, the disadvantages of existing implementations of describing the points of linking aspects and functional modules.

In the article [11] R. Laddad gives the basic concepts of AOP, introduces the term of through functionality, and also provides a brief description of the language AspectJ. The article [12] demonstrates the use of the aspect approach for implementing the protocol of interaction of objects in the implementation of design patterns, and also considers improvements in terms of modularity, increasing the degree of reuse and improving the perception of the source code of the classes of participants in the templates from the catalog.

M. Lippert and C. V. Lopes in their article [13] discuss general approaches to processing errors at the design stage of the software system, and offer tools for using AOP at different stages of the software lifecycle.

III. Features of the Aspect-Oriented Approach

In all programming languages, there are constructions that allow you to structure the description of the system as a hierarchical compositions of small modular elements. These constructions are aimed at identifying and constructing functional components that are expressed as classes, objects, modules, procedures, etc.

The disadvantage of functional solutions is that some concepts (tasks) such as synchronization, component interaction, stability, and security management that deal with multiple functional components can not be precisely localized. Typically, they are implemented by small code snippets, dispersed among many functional components. This last circumstance is at the core of the AOP. Aspect-oriented programming offers, in addition to functional components, the use of aspects that "cross" the functional components, and provides their composition in order to obtain high-quality software implementations. Examples of "intersecting" tasks (concepts) for aspects are synchronization, real-time constraints, error checking, parallel object interaction, stability, history tracking, protection, caching policy, profiling, monitoring, testing, etc. Some systems are characterized by very wide-reaching aspects. For example, component interaction, synchronization, remote call, parameter transfer strategies, load balancing, replication, fault neutralization, service quality, and distributed transactions are among the aspects of distributed systems [14].

Generally, any software system consists of several parts: the main (subject-oriented) and system parts, which carry the necessary functionality. For example, the core of the e-commerce system is designed to work with payments, while system-level functionality is designed to keep a log of events, security, performance, etc. Most similar parts of the system, known as crosscutting functionality [10], affect the many core object-oriented modules. You can consider a complex program system as a combination of modules, each of which includes in addition to business logic part of the through functionality from the set of requirements to the system.

The program consists of a set of implementations of modular concerns and crosscutting concerns, the latter, for example, security checks, implemented in the form of aggregates of dispersed code snippets in different program modules.

Crosscutting functionality combines such types of functionality, ideas, principles, methods, the implementation of which is fundamentally impossible in the form of only one hierarchy of interdependent modules, and also requires the insertion into the physically dispersed points of the program fragments of the new code (or the implementation of new code snippets in dispersed points of the target program, without their explicit insertion). Typically, code snippets implementing a new through-through functionality are executable designs (operators), often calls, but they can be descriptions (definitions) of data, also part of the implementation of through-the-line functionality.

With a high probability in any application you can find the sections of the through functionality that are available in the code under the guise of caching, logging, exception handling, transaction control, and delimitation of access rights.

Since the complexity of software systems is increasing, then there is a through functionality, that is, several approaches to address these problems. These approaches include mix-in classes [15], design patterns [16], and specific domain solutions.

When using mix-in classes you can put in them the implementation of crosscutting functionality. The components contain an instance of the mix-in class and allow other parts of the system to install their instance.

Software design patterns are effective tools for solving software design problems.

Specific domain solutions, such as frameworks and application servers, allow developers to make some crosscutting requirements to the level of these solutions. Specific domain solutions offer a specialized mechanism to address specific problems, but when technology changes, new approaches of solving the same problems have to be re-examined.

When developing a software system using existing programming technologies, crosscutting functionality will be included in all modules, resulting in a system that will be difficult to design, implement, and further support.

Aspect-oriented programming is one of the solutions that offers the means of allocating through-the-line functionality to individual program module-aspects in complex software systems.

In terms of AOP in the development of a fairly complex system, the programmer solves two problems: the development of components - the discovery of classes and objects that make up the subject area vocabulary; the development of services that support the interaction of components - that is, the construction of structures that provide the interaction of objects, in which the requirements of the task are fulfilled.

Modern programming languages (such as C ++) are focused, first of all, on solving the first problem. The component code is presented as a class, that is, it is well-localized, and, therefore, it is easy to view, explore, modify, reuse. On the other hand, when programming processes involving different objects, we get a code in which the elements associated with the support of this process are distributed by the code of the entire system. These elements are found in the code of many classes, their totality is generally not localized in the segment of the code. As a result, there is a problem of "confusing" code.

Within the AOP concept, it is argued that no design technology will help resolve this problem, the only way is to remain within the framework of component-oriented language. To program services that provide interaction of objects, you need special tools, possibly special languages. After the component and aspect coding phase in the corresponding languages an automated construction of the code optimized for execution (but not for viewing and modification) is performed. This final process is called merge or weaving.

Aspect-oriented approach considers the software system as a set of modules, each of which reflects a certain aspect-goal, the function of the system. A set of modules that form a program depends on the requirements of the program, the features of its subject area. Along with the functional requirements of the program are introduced and system-wide requirements, such as: integrity of transactions, authorized data access, logging events, etc. When designing a software system, the developer selects the modules so that each of them implements a certain functional requirement to the system. However, the implementation of some program requirements can often not be localized in a separate module within a procedural or object-oriented approach. As a result, a code that reflects such aspects of the functioning of the system will be encountered in several different modules. Traditional programming rules are used in the design of the program functional decomposition and do not allow the localization of crosscutting functionality in individual modules. The necessity of implementing crosscutting functionality by the available means leads to the fact that some component contains a code that reflects a lot of system requirements. This makes this module highly specialized, impairs the ability to reuse it, and in some cases leads to duplicate code. In turn, this increases the likelihood of errors, increases the time of debugging, reduces the quality of the program and to some extent complicates its maintenance. Aspect-oriented approach in some cases avoids the described problems and improves the overall design of the system, allowing the possibility of localization of crosscutting functionality in special modules-aspects.

The AOP allows the implementation of individual concepts in a slightly connected form, and, combining such implementations, forms a finite system. AOP allows you to build a system, using partitioning into separate modules (aspects) for implementing system-wide requirements.

The development within the AOP consists of three separate steps:

- aspect decomposition: breakdown of requirements for the allocation of general and crosscutting functionality. At this step, it is necessary to highlight the functionality for the modular level with the through-the-system-level functionality. For example, for the e-commerce system you can distinguish: creating a catalog of goods, a log of events, creating a list of products for sale;

- implementation of functionality: to fulfill each requirement separately. For the e-commerce system it is necessary to implement separately the module for forming the catalog of goods, the module of the event log, the module for creating a list of products for sale;

- layout of aspects: in this step, the aspect integrator defines the rules for creating its module-aspects, forming a finite system. In the example with the e-commerce system, it is necessary to determine, in terms of language, implement the AOP, when calling which operations it is necessary to make an entry in the journal, and upon completion of which actions it is necessary to report the success / failure of the operation.

AOP distinguishes a lot from traditional OOP approaches in the implementation of crosscutting functionality: a different way to imagine the process of decomposition, and the architecture of the resulting software product to a large extent goes beyond the concepts that are traditional for object programming. When designing on AOP, concepts are implemented completely independently of each other, as all existing links between them (crosscutting functionality) can be localized in aspect modules that describe the protocol of interaction of concepts. For example, in the module for creating a catalog of e-commerce products, there may be no log entry or a call to the module to create a list of products for sale, but when working, such a crosscutting functionality can be called if it is described in the interaction protocol. This is a serious step in the development of technology from object-oriented programming.

International Journal of Emerging Technology and Advanced Engineering
Website: www.ijetae.com (E-ISSN 2250-2459, Scopus Indexed, ISO 9001:2008 Certified Journal, Volume 11, Issue 05, May 2021)

Aspect in AOP is a system module, which has a crosscutting functionality. An Aspect Module is the result of an aspect decomposition, at which stage any phenomena, concepts, events that can be applied to a group of components obtained after an object decomposition are detected. Aspect represents a linguistic concept, similar to class, but only a higher level of abstraction.

Aspects can be applied to a large number of components and use so-called insertion points to implement regular actions that are usually dispersed throughout the text of the program. The aspect module describes the sections of the program implementation points, in which language instructions are embedded, which must be executed before, after, or instead of a strictly defined point of the program. Similar language instructions are functionality, which supports the interaction of components. In addition, the aspect modules can describe the role of components that can be affected by this aspect. In some implementations of AOP using aspect modules, you can influence the existing inheritance scheme. In terms of AOP aspect, it is a service that connects system components.

The automatic layout of aspects and the traditional component program modules is a key feature of the AOP, which determines the main advantage of this technology: it makes possible encapsulation of through functionality in individual software modules.

The automated layout of aspects and components is a powerful source of code generation and in general guarantees that the aspect will be applied to all component modules to which it relates, which is difficult to achieve if you implement the crosscutting functionality in the module (manually). Implementation of the automatic layout of aspects and components largely determines the possibilities of a particular aspect-oriented platform.

IV. E-COMMERCE SYSTEM AND LOGGING TOOLS

An e-commerce system is a system that has a modular structure and allows you to conduct sales procedures of certain types of goods (for example, books, software, computer and home appliances, clothing, footwear, furniture, office equipment, etc.) via electronic payment means (credit cards, smart cards, micro-payments, electronic money).

In the general case, within the e-commerce system, there is a certain Internet technology that provides the participants with the following capabilities: producers and suppliers of goods and services of various categories - to display products and services online, as well as to accept and process customer orders; customers (clients) - to view information (catalogs, price lists, etc.) about the offered goods and services, to make an order (requests, requests) and receive ordered goods (services) with standard browsers; manufacturers and suppliers - to accept payment, and buyers - to make payments using a payment system; in this case a bank becommes one of the participants of the system.

E-commerce systems are presented in two versions: the first one - in the form of the purpose of providing sales via the Internet; the second - in the form of complex decisions, integrated with the management of the enterprise.

E-commerce companies use different types and forms of business: e-shops, electronic stores, electronic exchanges, auctions, trading platforms, electronic advertising or travel agencies, electronic advisory centers, etc.

When developing an e-commerce system, you must adhere to the following rules:

1) the definition of the intended purpose of the system (will the system serve old or new clients; what the system is created for, for advertising or for sales, etc.);

2) creation of a demonstration version of the project and its presentation to potential users;

3) application of encryption mechanisms. Since e-commerce works with credit cards. This is usually a SSL (Secure Sockets Layer) technology, widely supported by web browsers and servers. However, it involves the storage of unencrypted information on the server, where hackers or simply dishonest employees can access it;

4) the establishment of interaction with related organizations. Despite the contradictions, there are common problems, such as increasing the level of information security when online orders are shipped;

5) providing the necessary data protection level.

To date, e-trading is the basis of e-commerce. The most common criterion for classifying an e-commerce company is the level of technology used to organize the trade process. In practice, trade systems are rarely fully automated and, by the degree of automation, such systems can be classified as electronic stores.

Logging - the process of recording information about events with some object (or as part of a process) events in the log / log file (for example, in a file). This is a chronologically ordered record of data processing operations.

For e-commerce, logging is very important. The system administrator can monitor user activity in the software system, analyze their preferences. Get information about the user's time spent in the system and his order.

To illustrate how to use the AOP, let's present the code to create the logging:

```
         * @Before("within(**)")
         */
        public function
beforeMethodExecution(MethodInvocation $invocation)
        {
           $file = 'log.txt';
             $obj = $invocation->getThis();
```

```
            if(is_object($obj) &&
get_class($obj)=="EShoppingCart" && $invocation-
>getMethod()->getName()=="getCost"){
    $person = $_SERVER["REMOTE_ADDR"]."
".date("d.m.y G:i:s")." User ".$this-
>getNameUser($_SESSION['name'])." added the book to
the cart.\n";
     file_put_contents($file, $person, FILE_APPEND |
LOCK_EX);
          };
          if(is_object($obj) &&
get_class($obj)=="EShoppingCart" && $invocation-
>getMethod()->getName()=="clear"){
            $person = $_SERVER["REMOTE_ADDR"]."
".date("d.m.y G:i:s")." User ".$this-
>getNameUser($_SESSION['name'])." formalized the
order.\n";
            file_put_contents($file, $person, FILE_APPEND |
LOCK_EX);
          };
    if(is_object($obj) &&
get_class($obj)=="BookController"){
if($invocation->getMethod()->getName()=="widget")
   {
    $data=$invocation->getArguments();
    if($data[0]=="zii.widgets.CDetailView" AND
$data[1]['attributes'][0]=="id"){
$person = $_SERVER["REMOTE_ADDR"]." ".date("d.m.y
G:i:s")." User ".$this-
>getNameUser($_SESSION['name'])." looked the goods
".$this->getName($_GET['id'])."\n";
    file_put_contents($file, $person, FILE_APPEND |
LOCK_EX);
      }
    if($data[0]=="zii.widgets.CListView")
         {
     $person = $_SERVER["REMOTE_ADDR"]."".
date("d.m.y G:i:s")." User ".$this-
>getNameUser($_SESSION['name'])." looked through the
list of all available offers.\n";
    file_put_contents($file, $person, FILE_APPEND |
LOCK_EX);
          }
        }
    }
```

V. Software Development Tools

To implement the e-commerce system, Yii Framework was selected. Yii is a highly effective, based on the component structure of the PHP framework, suitable for developing large web applications. It allows you to maximize the concept of reuse of the code and can significantly accelerate the web development process.

The name Yii means simple, effective, and extensible. Yii is a general-purpose web programming framework that can be used to develop virtually any web application. Thanks to its ease-of-use and advanced caching capabilities, Yii is especially suited for the development of high-traffic applications such as portals, forums, content management systems (CMS), e-commerce systems, and more.

Like most of other PHP-frameworks, Yii is an MVC framework. The advantage of Yii over other frameworks is its efficiency, broad capabilities and high-quality documentation. Yii from the very beginning is very carefully designed to meet all the requirements for the development of serious web applications. Yii is neither a byproduct of any project nor a collection of third-party solutions. It is the result of the extensive experience of authors in the development of web applications, as well as their research on the most popular web-based frameworks and applications.

Yii uses the Model-View-Controller design pattern, which is widely used in web programming.

MVC is aimed at separating business logic from the user interface so that developers can easily modify individual parts of the application without worrying others. In architecture MVC model provides data and rules for business logic, while the representation is responsible for the user interface (for example, the text, the input fields), and the controller provides interaction between the model and the presentation.

In addition, Yii also uses a front controller, called the application, which encapsulates the query processing context. The application collects information about the request and passes it for further processing to the appropriate controller.

Controller is an instance of a CController class or a class inherited from it. It is created by the application object when the user makes a corresponding request. When launched, the controller performs the appropriate action, which usually involves the creation of appropriate models and the rendering of the necessary representations. In the simplest case, the action is a method of a controller class, whose name begins with action.

The controller has the default action, which is executed when the user does not specify the action on request. By default, this action is called index. You can change it by setting the value CController :: defaultAction.

A model is an instance of a CModel class or a class inherited from it. The model is used for storing data or applicable business rules.

The model represents a separate data object. This can be a record of a database table or an HTML form with fields for entering data. Each field of a data object is represented by a model attribute. Each attribute has a label and can be validated using a set of rules.

Yii provides two types of models: the form model and Active Record. Both types are extensions of the base class CModel.

The form model is an instance of the CFormModel class. It is used to store data entered by the user. As a rule, we get this data, process it, and then get rid of it. For example, on an authorization page, a model of this type can be used to provide information about a username and password. For more details on working with forms, see Working with Forms.

Active Record (AR) is a design pattern used to abstraction access to a database in an object-oriented form. Each AR object is an instance of the CActiveRecord class or class inherited from it, and represents a separate row in the database table. The fields of this line correspond to the properties of the AR-object. See the Active Record section for details on the AR-model.

Representation is a PHP script that consists predominantly of user interface elements. It can cover PHP expressions, but it is recommended that these expressions do not change the data and remain relatively simple. According to the logic and representation separation concept, most of the logic code should be placed in the controller or model rather than in the submission script.

The representation has the name used to identify the presentation script file in the rendering process. The presentation name must match the name of the submission file. For example, to submit editing, the corresponding script file should be called edit.php. To display a view, you must call the CController :: render () method by specifying the name of the presentation. The method will try to detect the corresponding file in the directory protected/ views/ControllerID.

Inside the representation script, an instance of the controller is available via $this. Thus, we can refer to the properties of the controller from the representation code: $this->propertyName.

Extending the functionality of Yii is a standard practice in the development process. For example, to write a new controller, you need to expand Yii by inheriting its class CController; to write a widget - a class CWidget or a class of an existing widget. If the completed code is designed for use by third-party developers, we call it extension. Yii was originally designed in such a way that any part of it could be modified and supplemented for any needs.

To support the aspect-oriented approach, the library Go! AOP PHP was created. The main difference from all existing analogs is: a library does not require any PHP extensions. It does not use eval() and does not use the DI-container, it does not require a separate aspect compiler in the final code. Aspects are ordinary classes, which are organically using all the capabilities of the OOP. The library-generated code with interleaving aspects is very clean, it can be easily adjusted using XDebug, both for classes and aspects.

The most valuable thing in this library is that it is theoretically possible to connect it to any application, because to add a new functionality with AOP it is not necessary to change the code of the program at all, the aspects interwoven dynamically. For example: using ten to twenty lines of code, you can capture all public, protected, and static methods in all classes when you run a standard application and display the name of this method and its parameters when calling the method.

The main advantages of the Go! AOP PHP library:

- does not use PHP extensions, written entirely in PHP itself;

- can be used with any application in PHP;

- well-optimized (class caching, support for opcode caches)

- does not require a DI container to replace proxy objects;

- can intercept dynamic and static methods, methods in the final classes, as well as methods in the trades;

- can intercept access to public and protected fields;

- clean generated code; It is convenient to perform classes and aspects debugging using XDebug.

Adding aspects occurs only once, after which we get code from opcode-cache. Doctrine annotations are used for classes of aspects. Library Go! uses a unique Load-Time Weaving technology for PHP that allows you to track the load time of the class and modify this code before it parses the PHP. This allows you to dynamically modify the class code, without modifying the source code from the developers. At the beginning of the program, we initialize the core of the AOP, adding our aspects there, then transfer the management of the main program. When creating an object, an automatic boot of the class will work, which will determine the desired file name and try to download it. At this point, the call will be intercepted by the kernel, followed by a static code analysis and validation of the current aspects. The kernel will detect that there is a class in the code and that you need to implement the tips in this class, so the code transformers in the kernel will change the original class name, create a proxy class with the original name of the class redefinition method and give a list of tips for the given point. Next, PHP parser disassembles this code and downloads it into memory, with the name of the output class already having a class-decorator, so the usual method call will be wrapped up to the connection point with the connected tips.

When creating the e-commerce system, the following design methods were applied: a set of instructions (Advice) of a programming language executed before, after or instead of each of the points of execution (JoinPoint) included in the specified set (Pointcut); @Before and @After are a defined implementation order for a set that is executed before (after) execution; within (**) - global set; MethodInvocation contains an object (class for static method), method arguments, and information about the point in program.

VI. Conclusions

The emergence of new programming technologies has always been an interesting topic, since each new software development methodology allowed to solve the problems posed by its appearance, and significantly promoted Computer Science and the software industry as a whole. Complications of software systems as a global problem requires constant attention and study, so the appearance of AOP is a broad field for research with their subsequent practical application.

This article presents ways to use the aspect approach at different stages of the software lifecycle.

At the stage of designing the system, examples of implementation of such crosscutting requirements as implementation of the logging of events were given. The application of aspect-oriented decomposition at the stage of analysis and design allows improving the modularity of the system being developed - to highlight end-to-end functionality at different levels of abstraction and to localize it in individual modules-aspects. Such aspects are an integral part of the resulting software system. In this case, the components of the software system are completely free from the context of their application, and as a consequence, these components are fully ready for reuse. Based on the terms of modularity and components not burdened with code that are not related to the abstraction presented by it, it can be argued that the resulting cross-class composition looks more clear than in the case of object-oriented implementation.

At the stage of system development, tasks such as profiling, tracing, compliance with project agreements, tracking the accuracy of input and output data at different levels of abstraction are always addressed, the behavior of objects in a multithreaded environment is monitored, different approaches to the development of reusable components and strategies of their reuse. At this stage, the auxiliary aspects, as well as aspects that can later become part of the software system, and can provide significant help to the developer.

With regard to the difficulty of perceiving the source code, it can be said that the aspect approach increases modularity, which increases the ability of the components to be reused and increases the clarity of the code, which is not burdened with auxiliary built-in logic, which can subsequently be removed effortlessly.

At the stage of supporting the existing system, fewer architectural innovations are introduced, but more localized changes are used instead of taking into account new requirements and fixing old mistakes. The AOP provides new features with the support of software systems, with the addition of new functionality and the modification of existing ones.

An aspect approach provides new possibilities for integrating new requirements into a software system in comparison with the traditional object-oriented approach. If there are new requirements at the stage of system support, then such requirements are easily integrated without loss of modularity and modification of the code of components, which is very relevant at this stage of development.

Thus, the possibility of using an aspect-oriented approach is shown for tasks of a different nature. It should be noted that the main advantage of AOP is the improvement of the modularity of the software system, as well as the explicit identification of the structure of end-to-end functionalities, simplification of maintenance and making changes, the emergence of new possibilities for re-use of the code.

It should be noted that the AOP is not considered as a substitute for the established programming rules, but only serves as an extension that allows for end-to-end functionality.

The considered approach to the use of AOP for the e-commerce system offers great opportunities for the development of software systems. Thanks to the use of AOP, programming inherits all the benefits of aspect-oriented programming in the development of software systems: reducing the time, cost and complexity of development; simplify the maintenance of products and make changes to them; creation of reliable and safe systems through the allocation and implementation of crosscutting functionality in certain modules-aspects.

As further research, one can consider the work on formalizing the approach, as well as work to prove the correctness of the programs developed by the AOP. Also, one of the most promising researches in this area is currently considered the study of the event model of AOP and the applicability of this model for the construction of operating systems. In order for AOP, at last, to become an integral part of the process of developing and maintaining software systems, it is necessary, above all, to ensure the integration of AOP tools with traditional IDEs.

The main ideas and results included in the article, in particular, the development of a software system on the example of the system of electronic commerce, received a confirmation in the form of a certificate of registration of copyright to the work [17].

References

[1] G. Buch. Object-oriented analysis and design. Publisher Binom, Nevsky dialect, St. Petersburg, 1999. (in Russian).

[2] E. Dijkstra. A Discipline of Programming. — M.: Mir, 1978. – 277 pp. (in Russian).

[3] G. Kiczales, J. Lamping, A. Mendhekar, etc. Aspect-oriented programming. Published in proceedings of the European Conference on Object-Oriented Programming (ECOOP). Finland, Springer-Verlag LNCS 1241. June 1997

[4] Aspect-Oriented software development network www.aosd.net

[5] Homepage of the Subject-Oriented Programming Project, IBM Thomas J. Watson Research Center, Yorktown Heights, New York, http://www.research.ibm.com/sop/.

[6] K. Leiberherr. Component Enhancement: An Adaptive Reusability Mechanism for Groups of Collaborating Classes. In Information Processing'92, 12th World Computer Congress, Madrid, Spain, J. van Leeuwen (Ed.), Elsevier, 1992, pp.179-185.

[7] Ch. Simony. The Death of Computer Languages, The Birth of Intentional Programming, Microsoft Research, 1995, https://www.microsoft.com/en-us/research/publication/the-death-of-computer-languages-the-birth-of-intentional-programming/.

[8] Krzysztof Czarnecki, Ulrich Eisenecker Generative Programming: Methods, Tools, and Applications, Addison-Wesley, Paperback, Published June 2000.

[9] E.A. Zhuravliov, V.A. Kiryanchikov. About the possibility of dynamic integration of aspects in aspect-oriented programming. Izv. SPbGETU (LETI) Ser. Computer science, management and computer technologies. 2002 Issue 3. P. 81-86. (in Russian)

[10] E.A. Zhuravliov, V.N. Pavlov. About one approach to the implementation of Aspect-oriented programming. Izv. SPbGETU (LETI) Ser. Computer science, management and computer technologies. 2003/ (in Russian)

[11] Laddad, R. I want my aop!, part 1. JavaWorld. January 18, 2002. Avaliable at http://www.dmi.usherb.ca/~sgiroux/COURS/2008/ift785/fichiers/articles/AspectOrientedProgramming/jw-0118-aspect....pdf.

[12] O. Hachani, D. Bardou. Using aspect-oriented programming for design patterns implementation. Equipe SIGMA, LSR-IMAG, 38402 Saint Martin d'Heres Cedex, France.

[13] M. Lippert, C Videira Lopes. A Study on Exception Detection and Handling Using Aspect-Oriented Programming.Xerox PARCTechnical Report P9910229 CSL-99-1, Dec. 1999.

[14] Charnetsky K., Eiseneker U. Generative programming: methods, tools, applications. For professionals. - St. Petersburg: Peter, 2005. – 731 p. (in Russian)

[15] Bruno Schaffer Design and Implementation of Smalltalk Mixin Classes. Ubilab Technical Report 98.11.1 Universitдtsstrasse 84 CH-8033 ZurichSwitzerland.

[16] Myers G. Software Reliability. M.: Mir, 1980. (in Russian)

[17] A.s. Computer program «Information Web-system «Aspect e-commerce»/ Kucheryava O.M., Nabok D.A. – №59632; Application dated 17.032015; published from 013.05.2015 – 1p.