International Journal of Applied Information Systems (IJAIS) – ISSN : 2249-0868
Foundation of Computer Science FCS, New York, USA
Volume 5 – No. 9, July 2013 – www.ijais.org

A New Approach to Increase Information Systems Modularity using Event-Driven Software Architecture

Mohammad Masdari

Department of Computer Engineering, Science and research University, Urmia Branch, Urmia, Iran,

Esmail Amini

Computer Engineering Department Science and Research Branch,Islamic Azad University, West Azerbaijan, Iran

ABSTRACT
Software production line is one of the most active trends in software engineering. Most available methods are based on tripod architecture. In this paper, it is provided new frame with100% extensibility for data access layer. It is such a frame which can be used for all information system in all sizes. The difference of this method with the others is that the others are used to produce data access layer codes based on data model in the software production line. While it is suggested a main module as well as a tool based on designed patterns and innovations in this method which can be added to the available project and managed all the data access layer process and operation needless to produce any access layer to data for each project. By using this frame of data access layer, the projects are always ready and available and can be used in different projects without change.

Keywords
Software Architecture, Information Systems, Module Based System, Data Access Layer, Data Model

1. INTRODUCTION
Over time, the software systems are spreading and become more complicated. Extensibility and variability are the integral part of the software systems and the projects need them in each stage of development and even configuration and operation processes. Extensibility and variability in a system is one of the most important discussed challenges in software architecture. One of the main and important activities in software engineering is software production line. To do so, a lot of researches have been performing till now. The main goal is to provide software production line system which can produce a new software system based on beneficiaries' needs. There are a lot of methods and methodologies which perform widely to provide software production line system [6].

The software production line system issue is related to the new software system directly. It is used various architectures to provide it. Most available architectures are based on tripod architecture which derived from MVC model [1]. It is formed from Interface User (UI), Business Logic and data access layers [7]. The method we suggest to provide software production line for information systems is a Layer Based method. In this method, it is provided variable and extensible frames for the different layers of tripod architecture in which all the operations can be done and managed. As you can see in Fig (1), all Use Case of system are performed in Workflow Engine Framework. All the parts of the project will be used data accesslayer framework for connection and operation with data.

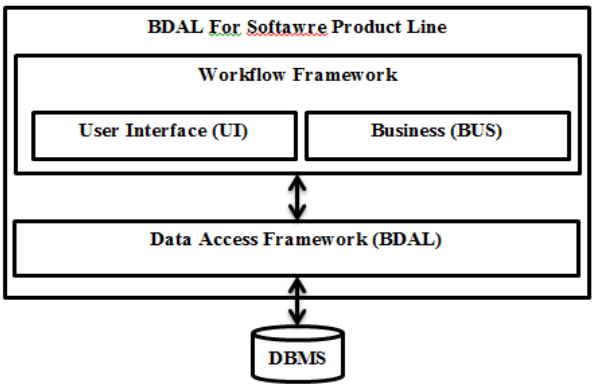

Fig 1: The proposed framework of information system in software production line

We discuss about BDAL data access layer to review and analyze the provided framework in this article. The steps propose to develop and expand framework of Data Layer will be as follow:

1- To review the earlier methods and tools and analyzing their extensibility and variability.
2- To review all operations and tasks of data access layer
3- To provide a Meta Model (architecture) for data accesslayer operation
4- To review the available classes in data access layer, identity and their attitudes
5- To provide a Meta Model (architecture) for data access layer class
6- To analyze the available Meta Models to Modules
7- To analyze and design the provided Modules
8- To test each modules (Unit Test)
9- To combine all modules to get the desired framework
10- To test the general framework (Integration Test)
11- To provide documents by using framework

2. DATA ACCESS LEYER, OPERATION AND TASK

In all kinds of available architectures for information systems esp. in those based on tripod architecture, we need an access layer to connect with system data [3]. All the operations related to the connection and operation commands of data base, insert and edit operation of all information of data base are administrated in this layer. According to Fig (2), regardless of any kind of methodology, Entity type classes which are placed in data access layer must support their data maintaining and validating, inserting, editing and deleting operation [3,8].

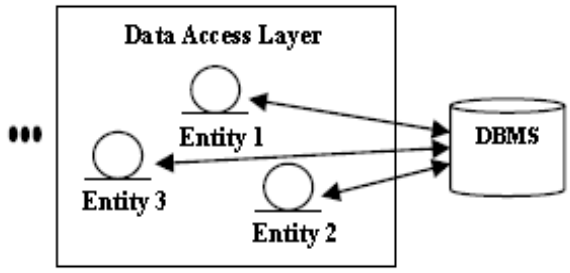

Fig 2: Entity type class of data access layer related to data

As a case study, an entity class which called Book can be indicated as Fig (3) in library management system.

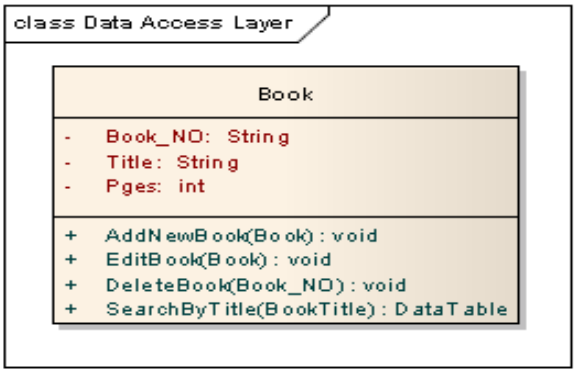

Fig 3: A sample of Entity class

In general, according to [3, 10], it can be provided all the data access layer activities or requirements of this framework as follow:

Req1: the ability to control the connection operation to data base

Req2: controlling Transactions and related errors.

Req3: the ability to do deleting, inserting and editing commands.

Req4: controlling cancelation review of all deleting, inserting and editing errors.

Req5: the ability to control cancelation review of all data of each entity.

Req6: the ability to review referential integrity errors and main key.

Req7: the ability to control the access level to data of each entity.

Each framework of data access layer must meet the noted requirements.

3. THE EARLIER METHODES OF DATA ACCESS LAYER MANAGEMENT

There are already many tools based on tripod layer architecture which involved in producing related codes of different layers. For data access layer, there are also various extensible tools which produced its data access layer based on data model of a system [11]. It can be noted to the tools such as Mai Generator, Subsonic, Oxygen Code generator, LLBN Gen Pro and so on. According to its usage, firstly, it must be provided the system data model based on system development stages and then by using this tool, the related codes of data access layer will be produced. The issues which must be reviewed are the input data validity, SQL orders production and its operation [4].

Most tools which produce data access layer consider these issues important. Then, they place parts in their tools to validation setting and SQL orders production and provide data access layer codes based on performed settings. The main problem of these methods is extensibility and variability. As after producing the first code, the data access layer codes are compiled and used. This status has very low variability and extensibility [2, 13]. For example, if we want to apply a new validity for an entity, we must reproduce and recompile the data access layer codes of that part. The other main problem is that there are changes in entities and attitudes of each of them. In this case, all the stages of data access layer codes production must be repeated. Some tools of data access layer production which apply capabilities in their tools provide considerable support to this problem as data access layer codes are changed in the direct relation with this tool. However, these methods can't solve the variability and extensibility problem [9, 15].

The major available problem in these methods is the lack of variability after system's configuration. For example, after full development of a system and its configuration, it we want to apply a new validation in the system, we have to repeat the production and compile stages of the code in which such a capability isn't exist at all in the available methods. Software production line theory which means providing software which can create different parts of codes of a project emphasize mainly on code production. However, code production doesn't solve the problem of variability and extensibility [2, 15]. The other main problem in code production methods is their security. For example, if we want to have the ability of deleting for an entity, we won't produce the related code and if we want to add it later, we will have code reproduction problem. It means that the capabilities can't be adjusted dynamically. The other problems include the production of many classes, lack of management and using software management principles. These methods will certainly meet the market needs. But, the ability of development, maintaining and variability will be their main problems [13].

4. BDAL FRAMEWORK

To produce an extensible, variable and dynamic framework, all related information of data access layer must be saved dynamically. It can be used the same data base to do so and save the data access layer information of each project along the data of that project. According the provided needs in

section 2 and 5, it can be divided data in 3 parts which must be saved in data access layer.

1- Meta Data: this part involves data related to the desired project's Meta Data and include the information of all tables, each one identity which recorded and maintained separately.

2- Validation: this part is proposed to save data related to all reviews which assigned to data access layer and each data.

3- Security: this part is proposed to save the access levels of each entity and their related data.

We indicate these 3 parts as MVS. To save these data, part of the related DBMS is used. To manage and control the related data of these 3 parts, it is used MVS Tools. According to Fig (4), the first level of provided framework consists of two main other modules namely Entity Manager and BD Manager.

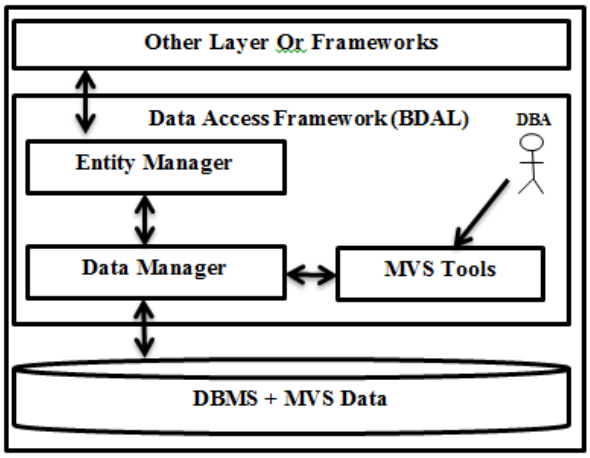

Fig 4: BDAL framework first level

To approve that the framework is capable of being implemented, we model the provided framework, design and analyze each module and provide each data model.

5. DB MANAGER MODULE

All connection operation to database and performing SQL commands are done in this module. As there are two different groups of data in system, it can be considered different classes for them to make easy the access control [6, 11]. For this purpose, we consider DB Manager as Fig (5) classes as follow:

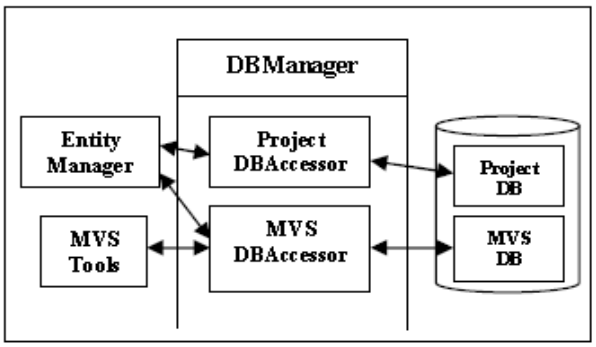

Fig 5: DB Manager Classes and their interaction

MVS Accessor class and Project DB Accessor are developed to interact and managed the related data of MVS and main project data of layer, respectively. Each available class in DB Manager can be connected to database and done data insertion and edition. The only difference is in the type of their accessible limitations. The identity and common behaviors of these two classes can be classified to higher class and two classes inherited from each other [14, 15]. If Base DB Accessor is called as father class, it would be included available behaviors and identity as Fig (6).

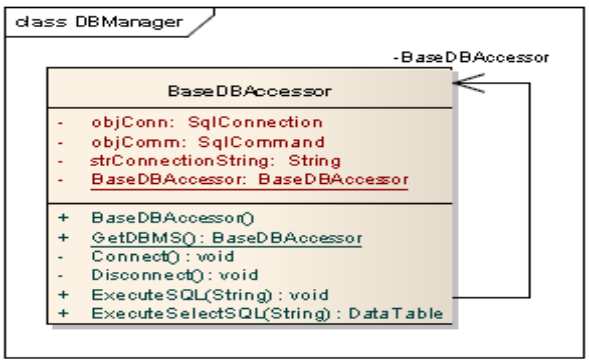

Fig 6: Designing Base BD Accessor class

The Execute SQL behavior will be to do Delete, Insert and Update commands which receive and perform an order as String. The possible errors are referred as Exception to recalled classes. Get DBMS behavior makes an object (element) from class and then turns it which is written based on Pattern Bridge [2] and supported a type of Connection Pooling for a user. Execute Select SQL behavior is to perform Select commands which returns its output as Data Table.

6. ENTITY MANAGER MODULE

As indicated in Fig (7), based on the tasks which must be done by this module, we broke down it to the other modules.

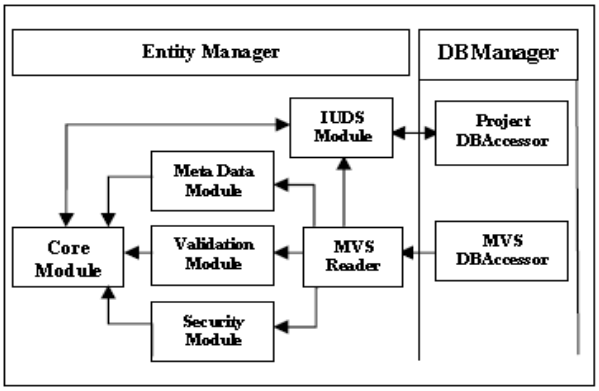

Fig 7: The internal architecture of Entity Manager Module

In fact, we use one full Entity rather several Entity classes which must be placed in data access layer for each entity. This full entity reads the related data of entities from MVS and performs the related operation. After that, we are going to analyze the available modules in the main module. After analyzing each module, Data Model of each of them will be provided. It is necessary to note that the goal of this article is to provide framework. Providing Class Model for each of them need full explanation which doesn't included in this article.

6.1 MVS Reader Module

As all related data of other modules in Entity Manager is recorded in database, large amount of data must be read per minute [6, 13].Accordingly, this module has been developed after entering all data related to Meta Data, Validation and Security in database which performed by data base manager and using MVS Tools. All data is read and maintained by this module in data base. The main goal of this module is to decrease connections to database and increase system performance. To maintain MVS data in this module it can be used Net Dataset or XML files. Apart from data maintenance method, this module must have classes which can reply other modules requested data related to MVS by using behaviors of each of them [2, 5]. The design stage of this module is performed after the full design of other modules.

6.2 Meta Data Module Data Model

In this module, the related Meta Data of developing project is recorded and maintained. According to Fig (8), all data of identities and tables are managed in it. To support Schema in database and as it is possible that the noted project consists of several modules, it is placed a class called Module. Each Module consists of several tables in which Tables' data such as the name and alias name of table in it. In Table class, all project Views are also recordable which can be recognized by Is View identity. Each Table consists of several Fields in which all of their data are also recordable. As the fields in different DBMS have different Data Type, it can also be recorded and managed all Data Type of DBMS.

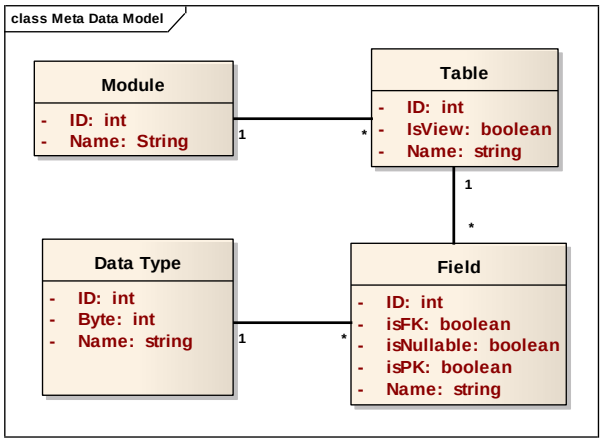

Fig 8: Meta Data of data model Module and its internal structure

6.3 Validation Module Data Module

In this module, all checking related to data are maintained and managed. The checking which performed in this module are just related to data access layer and doesn't include Business Rule. Accordingly, they are classified in two groups. The first group is Syntax Errors checking which reviews the lexical accuracy of input data and manages the related errors. It can be noted to different kinds of reviews such as to be numerical or chain and/or to be in a certain interval and so on. The second group of errors is related to the integrity of main and external keys which called Semantic errors. There aren't more than two reviews in this part. The first review is related to the main key. The repeated data isn't allowed to enter it. The other review is external key or referential integrity which must be controlled by this module [10, 11].

According to Fig (9), all related operations to Validation are managed in this module. Each field might have several Syntax Errors. For example, a field might have 3 kinds of checking which called "Is Not Empty"(review the field un-emptiness);"Is int"(review the field numerical status) and "Is In Range" (review the field valid interval). In all these three cases, it can be recorded a certain error and displayed to the user. In Method Name part, the name of the method in which the related checking is going to be done will be written. Based on this method, the checking operation can be managed. For each checking on a certain field, it can also be recorded the time of that error. Beside Syntax checking, it can be added two kinds of Semantic checking. It can be controlled main keys for the non-entry of repeated data and displayed a certain error for the occurrence of each of them. Meanwhile, if the key is external, we might record the table and identity of similar main key and display a certain error if the time of data insertion doesn't have a value in the main key. The considerable point is that even after a system configuration and its final delivery, it can be added different type of reviews of new errors on different fields without the need to recompile the code. In Validators part, all kinds of checking will be recorded. Then, if we need a new kind of checking, we can easily add it to the framework which indicates the extensibility of this framework to review errors [7, 14]. To do so, it is simply enough to write a new method for new checking, and then record its related data in Validators.

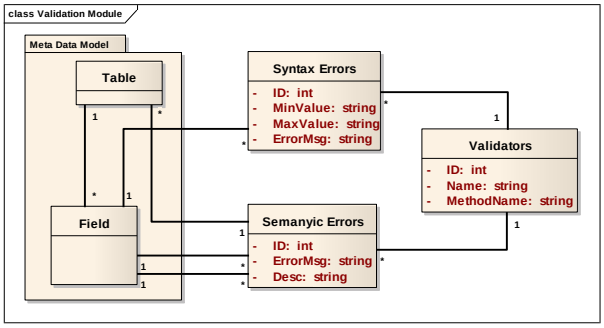

Fig 9: Data model of Validation Module

6.4 Security Module Data Model

In each project, data base manager can determine the access method to data. For example, based on certain circumstances, Delete isn't allowed for a table. In this case, it isn't placed a method called Delete or ability of Delete in Entity class in usual projects. For other orders, it is the same. Database manager of a project might determine allowed methods and capabilities on different data. In this module, data related to the access level of data of each table is managed. To complete this module, it must be done four main functions in each table. It means that different kinds of allowed Queries must be delivered to other layers programmers in methods framework. In this part, we just analyze Insert function. To Insert below data model, we recommend Fig (10). Accordingly, all Insert methods are manageable on table. For each table, it can be defined several Insert Commands in which we can determine data entry allowance to each identity and manage each one available errors. Each identity in each Insert command has a certain status which can be as allowed, record as Null and record as default [8, 9 and 16].

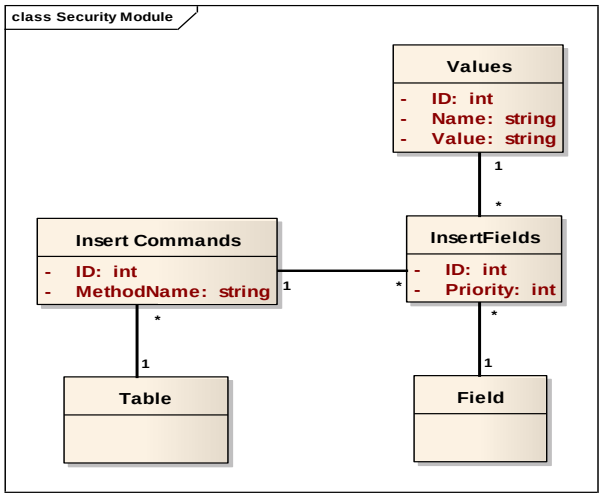

Fig 10: Security Module: Insert Data Model

For other SQL commands such as Update, Delete and etc, it can be provided such models.

6.5 IUDS Module

The goal of this module is to build dynamic SQL orders. As all data related to tables are available in MVS Reader, it can be obtained all related identities and data by determining the name of the table. It can be provided dynamic SQL commands by these data. Then, this module is linked to the project database by Project DB Accessor and performed SQL commands. If it fails, it will provide the considered error of desired order which is taken from MVS Reader. This error is transported to the other layers by Core [10, 15].

7. MVS TOOLS

To enter data related to Meta Data, Validations and security settings, it can be used tools which can receive desired data of database manager and record in MVS DB to use in Entity Manager. MVS tools are proposed to assign these tasks and goals and can be designed and produce once by designers and used in all software projects. Certainly, MVS Tools will consist of 3 main parts. Based on the provided data model in each part, data related to that part must be managed and recorded. It can be used many initiatives to provide the opportunity in which database manager can easily work with tools. For example, in the part related to system Meta Data, it can be used DBMS to get the project Meta Data. These data are available in schema-information of each DBMS. It is clear that MVS Tools might have facilities and features which can be used to dynamically produce all required data of BDAL and record in Meta Data framework and other settings. This helps that our proposed method plays its rule in system with maximum efficiency and without the need to recompile code [2, 3 and 6].

8. CASE STUDY: REVIEW THE PERFORMANCE OF THE PROPOSED METHOD IN LIBRARY MANAGEMENT SYSTEM

If such a framework has been fully developed, it can be find it in other layers as below. Here, we provide the necessary codes to record a sample book in library management system by using C#.Net language and proposed data access layer as below:

```
1: Entity Manager Entity=new Entity Manager ();
//Create New Object from Entity Manager Class
2: Entity. Set Entity ("Book");
//Entity .Set Entity (Entity Name);
3: Entity. Fields ["Book – No"] ="QA12";
//Entity .Fields [Field Name] = Value;
4: Entity. Fields ["Title"]"C#";
5: Entity. Fields ["Pages"] =250;
6: Entity. Insert ();
```

In the first line, as the first object is defined, all data related to MVS is read by MVS Reader from database. It is done once for the total project. We called this time ***Tf***. It includes connection to database (***T Connection***), reading Meta Data (***Tm***), Validation (***Tv***) and security (***Ts***) which can be generally indicated as ***Tf =TConnection +Tm+Tv+Ts***. In line 2, it is reviewed the availability of the table called Book via Meta Data module and if it isn't available, it will represent the considered error. It isn't done by database and performed by the read data from MVS Reader. Consequently, the time of theses reviews is a simple search which takes time ***O(Ct)*** and ***Ct*** is the number of project tables [11, 17].

In lines 3, 4and 5, the receiving operation and their checking are done which will based on MVS data. Such a review would be usually performed. Only in this case, the type of review is searched via MVS Reader which is equal to time ***Ca * O(Ce)*** in which ***Ca*** is the number of identities and ***Ce*** is the number of Syntax Errors table records. In line 6, it is reviewed firstly the access level of Insert Commands by security module and via calling Insert method. It has the time equal to ***O(Ci)*** in which ***Ci*** is the number of Insert Commands table records. It is necessary to point out that all data are taken from MVS Reader not from database. If it attains, data will delivered to IUDS module to insert. In that part, the related SQL command build is done and then followed by Insert operation. Building SQL command and its insert operation is occurred in usual status. Consequently, when this framework spent more than usual time in each Insert, it will be as below:

$$Tins = O(Ct) + Ca * O(Ce) + O(Ci) \qquad (1)$$

As ***Ca*** is a constant number and almost 20", the total time will be ***O(n)*** for each Insert. ***Tf*** time is also calculated once for total project. Due to the integrity of Update, Delete and Select commands and same as Insert method, it can be approved that for each of them, it would be spent the maximum time period (equal to ***O(n)***) which was more than usual time. Now, if we want to calculate additional time for a certain Use Case as the number of commands which figured in the Use Case are limited_ (for example, for the main Use Case, it can be about 10), so , we have more than usual time period (equal to ***O(n)***) for each Use Case [6,9 and 17].

9. CONCLUSION AND FUTURE WORKS

The models which are provided in previous parts, is an approval to extensibility and variability of provided framework. It is as we can apply necessary changes in each stage without any payment. The performance issue is reviewed and the only time equal to O (n) for each command takes more time than usual. The important fact is that due to

the nature of the quality features and trade off availability among them, this time period can be reduced but can't be removed. By developing and provided framework usage, the statistical data access layer is prepared for all projects and Time to Market projects are mostly decreased. It can also be optimized the software quality features in the provided framework by designing diagram class for each part and modules based on object-oriented design patterns.

10. REFERENCES

[1] Gomma, H., "Designing Software Product Lines Whit UML: From Use Cases to Pattern – Based Software Architectures", Addison Wesley, 2004.

[2] Greenfield, J., Short K., Software Factories: "Assembling Applications with Patterns, Models, Frameworks, and Tools", John Wiley & Sons, 2004

[3] Nock C., "Data Access Patterns: Database Interactions in Object – Oriented Applications", Addison Wesley, 2003.

[4] Crawford W., Kaplan J., "J2EE Design Patterns", O'Reilly, 2003.

[5] Lhotka R., "Expert C# 2008 Business Objects", Apress, 2009.

[6] Lhotka R., "Expert C# 2008 Business Objects", Apress, 2009.

[7] Y. Bao, X. Sun, K. S. Trivedi, "A Workload-Based Analysis of Software Aging, and Rejuvenation", IEEE Trans. Reliability,pp. 54-57, 2005.

[8] D. Chen, K. S. Trivedi, "Optimization for condition-based maintenance with semi-Markov decision process", Reliability Engineering and System Safety 90, 2005

[9] H. V. Ramasamy and M. Schunter, "Architecting Dependable Systems Using Virtualization", In Workshop on DSN,2007.

[10] Fähndrich, M., Aiken, M., Hawblitzel, C., Hodson, O., Hunt, G., Larus, J.R. and Levi, S., "Language Support for Fast and Reliable Message Based Communication in Singularity OS". In Proceedings of the EuroSys 2006 Conference, Leuven, Belgium,pp. 177-190, 2006.

[11] J. Gray, "Functionality, Availability, Agility, Manageability, Scalability—the New Priorities of Application Design", Proc. Int'l Workshop High Performance Trans. Systems, 2001.

[12] O. Etzion, P. Niblett, "Event Processing in Action, Manning Publications", USA, 2011.

[13] "Recommended Practice for Architectural Description of Software Intensive Systems". Technical Report IEEE P1471-2000, IEEE Standards Department, The Architecture Working Group of the Software Engineering Committee, 2000.

[14] A. Paschke, A. Kozlenkov, and H. Boley, "A homogenousreaction rules language for complex event processing," in InternationalWorkshop on Event Drive Architecture for ComplexEvent Process. ACM, 2007.

[15] K. Pope, "Zend Framework 1.8 Web Application Developmen"t, PACKT Publishing, 2009.

[16] G. Ahn, H. Hu, J. Jin. "Security-Enhanced OSGi Service Environments", IEEE Transactions on Systems, Man and Cybernetics—Part C: Applications and Reviews, Vol. 39, No. 5, 2009

[17] M. Fowler, D. Rice, M. Foemmel, E. Hieatt, R. Mee, R. Stafford. "Patterns of Enterprise Application Architecture", Addison Wesley, 2002.