World Academy of Science, Engineering and Technology
International Journal of Computer and Systems Engineering
Vol:1, No:3, 2007

Architectural, Technological and Performance Issues in Enterprise Applications

Melek Oktay, Ayşe Betül Gülbağcı, and Mustafa Sarıöz

***Abstract*—**Enterprise applications are complex systems that are hard to develop and deploy in organizations. Although software application development tools, frameworks, methodologies and patterns are rapidly developing; many projects fail by causing big costs. There are challenging issues that programmers and designers face with while working on enterprise applications. In this paper, we present the three of the significant issues: Architectural, technological and performance. The important subjects in each issue are pointed out and recommendations are given. In architectural issues the lifecycle, meta-architecture, guidelines are pointed out. .NET and Java EE platforms are presented in technological issues. The importance of performance, measuring performance and profilers are explained in performance issues.

***Keywords*—**Enterprise Applications, Architecture, Technology, Performance.

I. Introduction

ENTERPRISE applications identify the main components of organizations, information systems and how the components including staff, technology, business and resources work together to achieve business objectives [1]. Enterprise applications are very complex systems that are hard to design and implement.

Software development and software architecture have received much attention in the last decade even in highly respected big companies and small software firms in all over the world. The growing role of designing and organizing the system before coding is definitely covered and the importance of software architectures, design principles, design patterns etc. is understood exactly.

Many design tools, frameworks, design patterns are being developed for designing software systems but unfortunately still lots of projects fail because of many causes. The Chaos Report in 2004 [2] states that the project success rate is 34 percent of all projects. The project failure rate is 15 percent of all projects. 51 percent of all projects are over time, over budget or lacking critical features and requirements. According to the success and failure percentages, it can be indicated that it is very hard to achieve success in enterprise applications.

A large number of people at different backgrounds are involved in enterprise applications. There are complex businesses, management and technical issues which are difficult to control. Also it takes 6-7 years to complete an enterprise application from early design to successful company transformation [3]. Therefore, it is not unusual to have so many problems like over-budgeting, over-time in enterprise applications. There are many pitfalls, bottlenecks and confusing works from beginning to end of an enterprise application.

In this paper we propose to analyze the significant issues in enterprise applications. In enterprise applications, naturally there are lots of things to consider due to the complexity of the system but it is impossible to cover all of the issues. Therefore; we will present the architectural, technological and performance issues in enterprise applications and give some recommendations to overcome the common problems in enterprise applications.

This paper is structured as follows: In Section 2, the architectural issues in enterprise applications are presented. In Section 3, the technological issues including Java EE and .NET platforms are pointed out and the frameworks in Java EE are mentioned. In Section 4, the performance issues are presented.

II. Architectural Issues

Architectural issues are defined in the phases of software lifecycles. The software lifecycle is an abstract representation of software process that defines the software development strategy, steps, methods, activities and product of a software application.

Developing an enterprise application begins with analyzing and organizing the elements according to the requirements and sources. Then comes, designing the system, defining the true architecture and implementing the architecture in high quality. The traditional lifecycle phases of a software application are shown in Fig. 1.

Manuscript received March 10, 2007.

Melek Oktay is with the Computer Engineering Department, Fatih University, Istanbul, 34500 Turkey (corresponding author to provide phone: +90-2128663300-5520; fax: +90-2128890906; e-mail: moktay@fatih.edu.tr).

Ayşe Betül Gülbağcı is with the Computer Technologies and Programming Department, Vocational School, Fatih University, Istanbul, 34500 Turkey (e-mail: betule_mail@yahoo.com).

Mustafa Sarıöz is with the Computer Engineering Department, Fatih University, Istanbul, 34500, Turkey (e-mail: msarioz@fatih.edu.tr).

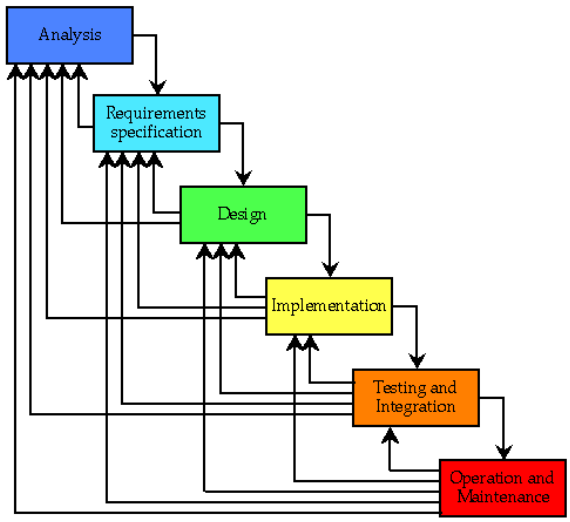

Fig. 1 Software lifecycle

The first step of a good architecture is well done analysis. In the beginning phases, organizational units and business functions to be supported by the system are defined. Also the technical environment and draft project plans are described.

In requirement analysis phase, lack of understanding and communication with customer is an important pitfall that affects the architecture of enterprise applications. It is impossible to think through all the issues that users need properly; but it is important to understand the requirements correctly. The necessities of users change continuously and this causes redesigning the architecture. Designing the system according to the wrong/changing requirements causes headaches [4] in most of the enterprise applications. If a project deviates too far away from original specifications and does not meet the user requirements, it fails because of being late or over-budget. The solution is communicating the users more often and to get requirements correctly.

Making the right architectural decisions is very important. Today's software can be legacy system of tomorrow; therefore good architectures are needed [5]. When beginning a project, the decision of using an existing architecture/framework or designing a new one must be made. Architectural decisions are made according to the requirements at different levels. Since architecture is the structural elements of the system together with their externally visible properties and relationships**,** high level and low level decisions must be made.

High level decisions are related with the integrity and structure of the system which is called "meta-architecture". Meta-architecture involves style, patterns of composition or interaction, principles, and philosophy, rules certain structural choices out, and guides selection decisions and trade-offs among others [6]. As seen in Fig. 2, architecture is the middle layer and by taking care of meta-architecture, architectural diagrams and system priorities are formed. In low-level architecture, architectural guidelines and policies are decided by using design patterns, frameworks, infrastructure and standards.

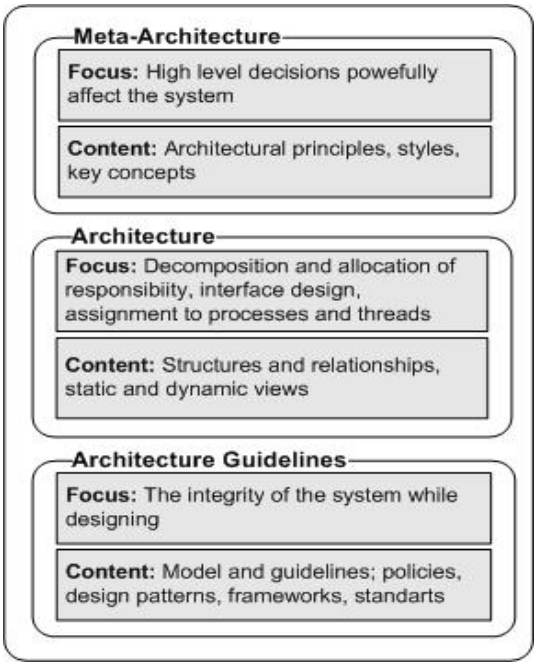

Fig. 2 Architecture

Software architecture should be designed well and it should be supported with design patterns, reusable class libraries that allow great flexibility for the project [28]. Especially design patterns are elegant solutions for common problems and they provide usability to architecture.

Before the enterprise software architecture is designed, some of the existing architectural frameworks such as MVC [13], PCMEF [11] and XWA [10] should be analyzed; because the appropriate framework simplifies the architecture.

MVC is Model–View-Controller paradigm that separates View from Model. Model is the non-visual object that consists of application data. View is responsible from showing Model data in a user interface. Taking input from user, managing the model and updating View is a Controller's responsibility [13].

PCMEF is layered paradigm that consists of presentation, control, domain, domain and foundation layers [11]. The aim of PCMEF is minimizing package coupling, decreasing dependency and increasing stability with using downward dependencies (higher layers depends on lower layers). When upper layers are changed, lower layers are not affected; this provides loose coupling and allows programmer to build roundtrip architectural modeling [28, 11, 29, 30].

XWA (Extensible Web Architecture) is based on MVC and PCMEF combines the advantages of these frameworks [31].

III. Technological Issues

Enterprise applications are developed with development platforms. A development platform includes programming language/s, run-time environment, and reusable class libraries. Using the right application development platform is very significant. If there exist troubles related to the development platform, changing the development platform may be expensive or impossible.

It is a well known fact that most of the computing platforms are Turing Complete, so software developed with one platform can be developed with other platforms in theory. Although the development platforms facilitate solutions of the problems, accumulations of one platform may be more improved than the others. Therefore, selecting the true platform is considerable for the success of the project.

There are two leading technologies in enterprise-level application development: .NET [16] and Java EE [17]. In addition, there are alternative technologies like WebObjects [18], Coldfusion [19], and PHP (Hypertext Preprocessor) [20].

.NET is a Microsoft product described as the software that connects information, people, systems and devices. Java EE is a set of specifications for developing enterprise-level applications, created by the Java Community Process (JCP).

We will analyze .NET and Java EE platforms according to dependency, vendor, usage of web services, cost and security.

.NET is a language independent platform that allows programmers to use different properties of programming languages such as C#, VB.NET J# (Java for .Net) etc. Besides, .NET is tied closely to the Windows operating system. It is possible to say that if Windows-only environment is being used then Microsoft.Net provides good solutions for enterprise applications (with limited choice and limited influence on future directions but the benefits of one source and a known supplier) [21].

Java EE is platform independent that runs on any operating system. However, only java can be used as a programming language. This property of Java EE provides an advantage in heterogeneous environments that include different platforms [21].

Also one of the main advantages of Microsoft.NET is its integrated support for web services. Java Platform achieves this with many components [22]. Since Java has the disadvantage of being developed long before Web Service Standards are set, there is not an integrated architecture for web services in Java EE. However Sun has taken aggressive steps to incorporate Web Services into the Java EE standard [21].

Microsoft is a proprietary product based on unpublished standards with valuable costs; but Java EE has an advantage about multi-vendor support, which includes commercial and noncommercial solutions.

Security is an important issue that every enterprise project must satisfy. Security is a comprehensive phenomenon which includes secure communication, access control, user authentication, auditing and tracking etc... .Net and Java platforms have strengths and weaknesses about security.

.NET utilized the past experience of Java EE, so while java evolves its security capabilities gradually, .NET incorporated more security capability into its original design. Because of simpler and clearer design, .NET provides advantages in security and scalability. Another important point is the Internet Information Server (IIS) that is the web server of .NET. It is one of the most attacked server software in the world [23].

Table I summarizes the properties of .NET and Java EE explained above.

TABLE I
JAVA EE AND .NET

	.NET	Java EE
Dependency	Platform dependent Language independent	Platform independent Language dependent
Vendor	Microsoft	30+
Web services	Integrated support for web services	Web services is retrofitted to Java by APIs, Have wider choices
Cost	A proprietary product Has some valuable cost	Commercial and non-commercial multi-vendor support
Security	Simpler and clearer security design IIS is the most attacked server	Complex security design

Java EE and .NET Enterprise applications are generally considered as multi-tiered applications that consist of three tiers: application (client) tier, middle tier and EIS (Enterprise Information System) tier. The tiers in Java EE and .NET are illustrated in Fig. 3.

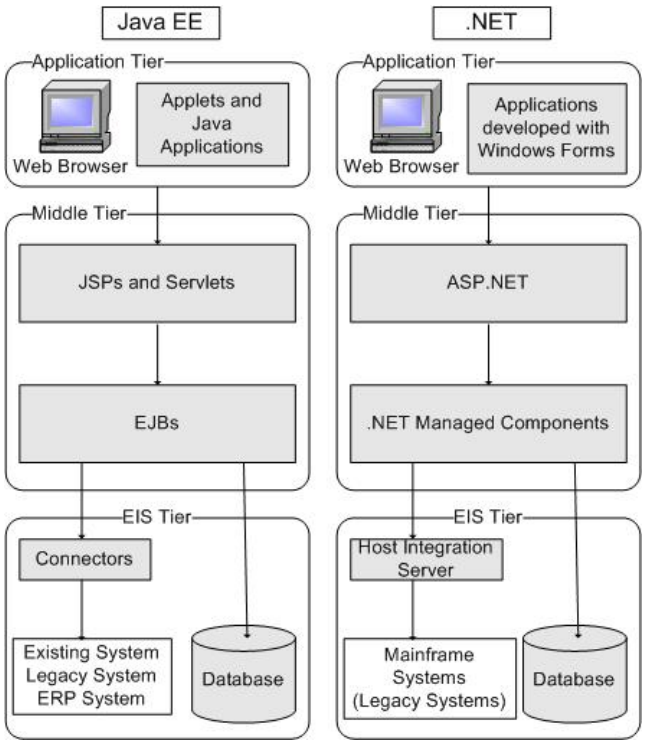

Fig. 3 Java EE and .NET tiers

The application tier is the client side of application. In Java EE, client tier includes java applets, web browsers and java applications. In .NET, there are applications developed with Windows forms and web browsers in client tier.

The second tier is the middle tier, which is divided into two parts: web tier and business tier. In web tier, Java EE uses JSP and Servlets, but .NET technology offers ASP.NET. Business tier performs data logic and business processing which are the core functionalities of an application. In Java EE, business

code is handled with Enterprise Java Beans (EJB). .Net offers .NET Managed Components as a business tier component of enterprise architecture.

The third layer is EIS tier, which consists of database servers, enterprise resource planning systems, and other legacy data sources, like mainframes. Most of the large corporations have existing codes. Both of the technologies offer solutions for legacy integration. .NET uses Host Information Server to connect to legacy systems; but Java EE offers Java EE Connector Architecture (JCA) for integration.

An application development platform may have complex technologies that are hard to use. For this reason, it is essential to have different alternatives for the programmers In Java EE many frameworks are developed in each tier to simplify the complex processes and decrease the application development period. In the subsection below the frameworks in Java EE are briefly described.

A. Frameworks in Java EE

In Java EE there are many third-party products and open source frameworks. In this section, we will briefly present the third-party products and frameworks used with Java EE. There are different technological choices in Java EE becoming popular in recent years. The open source frameworks and third party products in each tier are shown in Fig. 4.

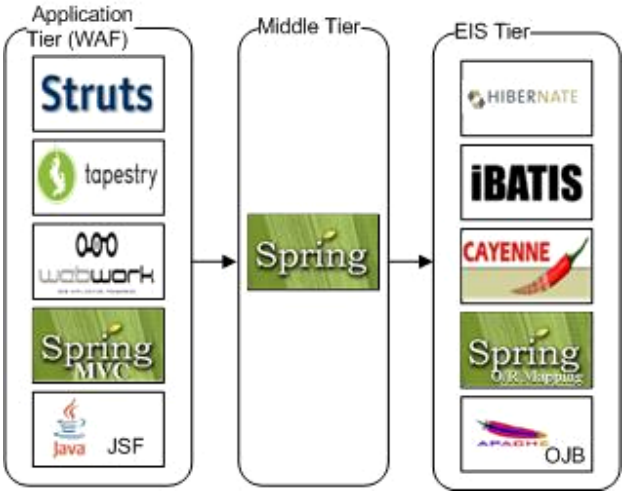

Fig. 4 Java EE frameworks

In the last couple of years, many Web Application Frameworks (WAFs) are developed with Java. WAF is a reusable, skeletal, semi-complete modular platform that can be specialized to produce custom web browsers via Http(s) protocol [9]. WAF applies Model-View-Controller design pattern to web applications and it is typically in the Model 2 architecture, so it can be separated from presentation tier with application logic. It makes software development and team organization simpler. Therefore, these frameworks facilitate software development and reduce the amount of time and effort significantly. JSF [32], Struts [33], Tapestry [34], Webwork [35] and Spring MVC [36] are examples of WAF. Taxonomies of these frameworks are described in [9]. These kinds of Web Application Frameworks are fully developed in Java. Briefly, these frameworks make Java Platform more practical and applicable.

In the middle tier, Enterprise Java Beans (EJB) technology is used. Although the EJB is good for transactional processing [24], it is very complex [13, 15, 24]. The complexity of EJB makes using it difficult for the reasons below [15]:

- It makes application harder to test.
- It makes harder to deploy application.
- EJB makes simple things harder.
- Reduced choice of application servers.

Programmers of Java EE platform can use Spring Framework [25] which is a good alternative of EJB. Spring Framework is not as complex as EJB because of its lightweight container architecture. The lightweight container architecture makes Spring run outside of EJB container, so Spring can run on a simple Servlet Container such as Tomcat. The Spring Framework makes application testing easy, and also decreases the complexity of application server administration and allows programmers greater portability [26].

EIS tier includes Object Relational Mapping (ORM). ORM is a programming technique that converts data to incompatible type systems in databases and object-oriented programming languages. Most of the enterprise projects use database and the developed software run with the database. The software developed always has to convert application data to database entity and database entity to application data with converters. For this reason, programmers should implement their converters but this takes so much effort and time. The ORM frameworks are developed to meet necessity of converters such as Hibernate [37], iBatis [38], Cayenne [39], Spring ORM [36] and Apache OJB [40].

IV. Performance Issues

Performance has an important role especially in real world enterprise applications and it often determines the success or failure of enterprise applications [6].

However it is difficult to make decisions about performance from just looking at the design. Rather, people have to actually run the code and measure performance [7]. If programmers have not been satisfied about performance and want to optimize code for improving performance, they must have real evidence about it. It is important to make the performance tuning in early cycles of the project. If there are architectural mistakes or dangerous bottlenecks, it is beneficial to catch them in early cycles of the project and redesign the architecture.

For getting real evidence and ensure level of performance, programmers should use some benchmarks such as web-load testing tools (Microsoft Web Application Stress Tool (WAS), Apache JMeter, etc...). These tools create multiple connections to the web application like real applications in production. Purpose of this test is to measure or observe behaviors of a web application when it is in production, and also find any bottlenecks if there are. For this reason, the application should run on production hardware or the closest available hardware to production hardware [8]. In addition to

this, configuration of application should be the same as production level such as logging level, framework configuration, VM configuration, etc.

If any bottleneck is found in the application, a profiler could help to determine which method or methods are the reasons. Then, the method is found and optimized. If an application meets performance and throughput requirements, people should not spend much time with profiling or optimizing.

Profiling can indicate which pieces of slow code matter. A profiler can help programmers to find the slow methods even in thousands of codes so programmers do not need to worry about and also computers are much better at that kind of task than humans [6]. In addition to find slow codes in application, profiling can be helpful for understanding dynamic behaviors of code for example, unnecessary call method or accidentally calling method twice. A good profiler can show number of objects of each class being created.

V. Conclusion and Future Work

Enterprise architectures are very complex information systems which involves many people from different backgrounds and different business, management and information processes. From the first phase to last phase of software development, there are many important points which should be concerned. In this paper, we focused on architectural, technological and performance subjects which are essential to develop supportable and extensible enterprise applications.

The important factors in the problem analysis, requirement analysis and design phases that affect the software architecture are summarized. The significance of architectural decisions is also mentioned. The meta-architecture concept involves high level decisions such as structural decisions, principles, and philosophy of software. Some architectural frameworks such as MVC, PCMEF and XWA should be used.

In technology issues, we have considered available platforms for enterprise applications. Solution domain is directly related to the platform used. If environment is heterogeneous (such as Windows, Linux and Mac OS X), it is appropriate to use Java EE platform. And also multi-vendor support, third-party product and open source projects increase productivity and efficiency of Java EE. However, if the solution domain platform is windows-only, .NET is a feasible platform. The architecture of enterprise applications is divided into three tiers which are application (client) tier, middle tier and EIS (Enterprise Information System) tier. In Java EE, there are many third-party products and open source frameworks for each tier. This provides alternatives to programmers while implementing an application.

Performance determines the success or failure of a project. First, efficient and suitable software architecture should be chosen. If architecture is designed well and it is supported with design patterns, reusable class libraries allow great flexibility for the project. If a project does not meet performance requirements, the whole project goes for nothing and it fails. Determining the performance is not feasible by just looking at the code; to have real evidence about performance the code should be run to get the right performance.

As future work, we will implement an enterprise web application by following the architectural, technological and performance issues presented in this paper. We propose to use open source third party Java EE frameworks (Spring, JSF and Hibernate) and implement a enterprise application to be used in municipalities.

References

[1] S.H. Kaisler, F. Armour, M. Valivullah, "Enterprise Architecting: Critical Problems", *IEEE Proceedings of the 38th Hawaii International Conference on Systems Sciences*, 2005, pp. 224b.

[2] The Standish Group. Available: http://www.standishgroup.com

[3] P. Booth, Z. Matolcsy, B. Wieder, ERP Systems Survey Benchmark Report, 1999. Enterprise Resource Systems Project, University of Technology, Sydney.

[4] M. Fowler., "Is Design Dead?", *Software Development Magazine*, Nr. 4, Apr. 2001.

[5] S. Tilley, "Five Year of Web Site Evolution", *5th IEEE International Workshop on Web Site Evolution*, pp. 103- 107, 2003.

[6] R. Malan and D. Bredemeyer,(2002). "Software Architecture: Central Concerns,Key Decisions". Available: http://www.bredemeyer.com/pdf_files/ArchitectureDefinition.PDF

[7] IEEE Standard 1061-1992, *Standard for Software Quality Metrics Methodology*, New York: Institute of Electrical and Electronics Engineers, 1992.

[8] IISO/IEC 9126-1, Software Engineering - Product Quality - Part 1: Quality Model, 2001.

[9] T. C. Shan, W. W. Hua, "Taxonomy of Java Web Application Frameworks," in Conf. Rec. 1995 IEEE Int. Conf. on e-Business Engineering, pp. 378–385.

[10] L. Madeyski and M. Stochmial/ek, "Architectural Design of Modern Web Applications," Foundations of Computing and Decision Sciences, vol. 30, no. 1, pp. 49--60, 2005. [Online]. Available: http://madeyski. e-informatyka.pl/download/23.pdf

[11] L. A. Maciaszek, B. L. Liong, *Practical Software Engineering,* Addison Wesley, 2004.

[12] R. Johnson, *Expert one-on-one J2EE Development without EJB*, Wrox, 2004.

[13] M. Fowler, *Patterns of Enterprise Application Architecture*, Addison-Wesley, 2003.

[14] R. Johnson, *Expert one-on-one J2EE Design and Development*, Wrox, October 2002.

[15] Microsoft .NET platform http://www.microsoft.com/net/default.mspx

[16] Java Platform, Enterprise Edition, http://java.sun.com/javaee/index.jsp

[17] Apple WebObjects , http://www.apple.com/webobjects/

[18] Adobe ColdFusion, http://www.adobe.com/products/coldfusion/

[19] PHP, http://www.php.net/

[20] A. Aitken, "An Overview and Comparison of Three Major Enterprise Application Development Platforms*", in Conf. Rec. 2005 IEEE Int. Conf. Industrial Informatics***,** pp. 268-27**4,** 2005.

[21] S. Kachru, E. F. Gehringer, "A Comparison of J2EE and .NET as Platforms for Teaching Web Services," *34th ASEE/IEEE Frontiers in Education Conference*, October 2004.

[22] G. Kunene:, "*Software Engineers Put .Net and Enterprise Java Security to the Test*", http://www.devx.com/enterprise/Article/16823/

[23] T. Neward, *Effective Enterprise Java*, Addison-Wesley, 2004.

[24] R. Johnson, J. Hoeller, A. Arendsen, T. Risberg , C. Sampaleanu, "*Professional Java Development with the Spring Framework* ", Wrox, 2005.

[25] J. Arthur, S. Azadegan, "Spring Framework for rapid open source J2EE Web Application Development: A case study", *IEEE 1st AGIS Conference*, 2005.

[26] R. Johnson, "J2EE Development Frameworks", *IEEE Computer*, Vol.38, 2005.

[27] E. Gamma, R. Helm., R. Johnson, J. Vlissides, *Design Patterns, Elements of Reusable Software*, Addison Wesley, 1995.

[28] L. A. Maciaszek, Roundtrip Architectural Modeling, *Conf. in Research and Practice in Information Technology Series,* Vol. 107, 2005.

[29] L. A. Maciaszek, Developing Supportable Enterprise Information Systems – Architectural, Managerial and Engineering Imperatives, *Int. Conf. on Software Maintenance,* pp. 721-722, 2005

[30] L. A. Maciaszek, "Developing Supportable Enterprise Information Systems – Architectural, Managerial and Engineering Imperatives," *Proceedings of the 21st IEEE Int. Conf. on Software Maintenance,* pp.721-722, 2005.

[31] L. Madeyski and M. Stochmialek, "Architectural Design of Modern Web Applications," *Foundations of Computing and Decision Sciences*, vol. 30, no. 1, pp. 49--60, 2005.

[32] Java Server Faces**,** http://java.sun.com/javaee/javaserverfaces/

[33] Apache Struts, http://struts.apache.org/

[34] Apache Tapestry, http://tapestry.apache.org/

[35] Webwork ,http://www.opensymphony.com/webwork/

[36] Spring Framework , http://www.springframework.org/

[37] Hibernate, http://www.hibernate.org/

[38] Apache iBatis, http://ibatis.apache.org/

[39] Apache Cayenne, http://cayenne.apache.org/

[40] Apache OJB, http://db.apache.org/ojb/