

PRODUCTIVITY ORIENTED PROGRAMMING FRAMEWORK

A THESIS SUBMITTED TO
THE GRADUATE SCHOOL OF NATURAL AND APPLIED SCIENCES
OF
ÇANKAYA UNIVERSITY

BY

OZAN ALİ KAYA

IN PARTIAL FULFILLMENT OF THE REQUIREMENTS
FOR
THE DEGREE OF MASTER OF SCIENCE
IN
COMPUTER ENGINEERING

JULY 2009

Title of the thesis: **Productivity Oriented Programming Framework**

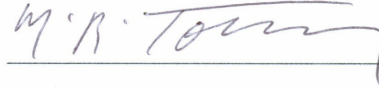
Submitted by **Ozan Ali Kaya**

Approval of the Graduate School of Natural and Applied Sciences, Çankaya University



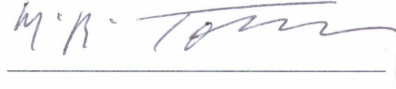
Prof. Dr. Yahya K. Baykal
Acting Director

I certify that this thesis satisfies all the requirements as a thesis for the degree of Master of Science.



Prof. Dr. Mehmet R. Tolun
Head of Department

This is to certify that I have read this thesis and that in my opinion it is fully adequate, in scope and quality, as a thesis for the degree of Master of Science.



Prof. Dr. Mehmet R. Tolun
Supervisor

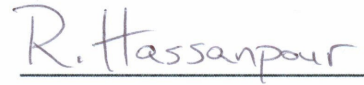
Examination Date : 13.07.2009

Examining Committee Members:

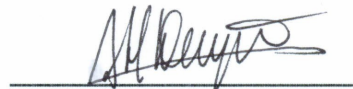
Prof.Dr. Mehmet R. Tolun (Çankaya Univ.)



Assist. Prof. Dr. Reza Hassanpour (Çankaya Univ.)



Assoc.Prof.Dr. Ali Doğru (METU)

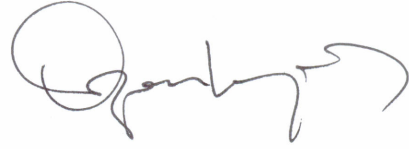


STATEMENT OF NON-PLAGIARISM PAGE

I hereby declare that all information in this document has been obtained and presented in accordance with academic rules and ethical conduct. I also declare that, as required by these rules and conduct, I have fully cited and referenced all material and results that are not original to this work.

Name, Last Name: OZAN ALI KAYA

Signature :

A handwritten signature in black ink, appearing to read 'Ozan Ali Kaya', written in a cursive style.

Date : 13.07.2009

ABSTRACT

PRODUCTIVITY ORIENTED PROGRAMMING FRAMEWORK

Kaya, Ozan Ali

M.S., Department of Computer Engineering

Supervisor : Prof. Dr. Mehmet R. Tolun

July 2009, 43 pages

In this thesis, the basic problems that are faced during software application development are mentioned and the affects of these problems over the productivity of software developers are discussed.

The infrastructure that is needed to provide productivity is implemented with the framework called Productivity Oriented Programming Framework, which makes it possible for developers to produce qualified products faster and more effectively with no extra effort.

Keywords: Productivity, Productivity Oriented Programming, Service, Service Oriented Architecture

ÖZ

ÜRETKENLİĞE YÖNELİK PROGRAMLAMA UYGULAMA ÇATISI

Kaya, Ozan Ali

Yüksek Lisans, Bilgisayar Mühendisliği Bölümü

Tez Yöneticisi : Prof. Dr. Mehmet R. Tolun

Temmuz 2009, 43 sayfa

Bu tez çalışmasında, yazılım ürünlerinin hazırlanmasında karşılaşılan temel sorunlar belirtilmiş ve bu sorunların yazılım geliştiricilerde üretkenliği nasıl etkilediği tartışılmıştır.

Üretkenliğin sağlanması için gerekli altyapı, Üretkenliğe Yönelik Programlama Uygulama Çatısı adı verilen örnek yapı tasarlanarak gerçekleştirilmiştir. Bu altyapı kullanılarak hızlı ve kaliteli ürün hazırlamaya olanak sağlanmıştır.

Anahtar Kelimeler: Üretkenlik, Üretkenliğe Yönelik Programlama, Servis, Servis Tabanlı Mimari

ACKNOWLEDGMENTS

I would like to express my special thanks to my supervisor Prof. Dr. Mehmet R. TOLUN for his supports and suggestions.

I would also like to thank my love, Esra for her labour and self-sacrifice throughout the years.

TABLE OF CONTENTS

STATEMENT OF NON-PLAGIARISM PAGE	iii
ABSTRACT	iv
ÖZ	v
ACKNOWLEDGMENTS	vi
TABLE OF CONTENTS	vii
LIST OF FIGURES	viii
LIST OF ABBREVIATIONS	ix
CHAPTERS	
1 INTRODUCTION	1
2 PRODUCTIVITY IN SOFTWARE	4
2.1 Persistency	4
2.2 Reporting	5
2.3 Logging	5
2.4 Organization and Authentication	5
2.5 Workflow	6
2.6 Scheduling	6
2.7 Date, Time and Exchange Rate Operations	6
2.8 Multilanguage Support	7
2.9 Cache System	7
2.10 File Server	7
2.11 Third Party Application Integration	7

3	CURRENT STUDIES USING JAVA ANNOTATIONS TO INCREASE PRODUCTIVITY	9
3.1	jpa2web	9
3.2	OpenXava	10
3.3	Seahorse	10
3.4	Stripes	10
3.5	Spring Framework	11
3.6	JBoss Seam	11
4	ARCHITECTURE AND CORE LAYER OF POPF	12
4.1	POPF Implementation : Business Server	14
	4.1.1 Service Production	15
	4.1.2 Report Preparation	21
5	SAMPLE REQUIREMENT LAYER FOR POPF: ORGANIZATION AND AUTHORIZATION SYSTEM	32
6	CONCLUSION	41
6.1	Future Works	41
	REFERENCES	R1

LIST OF FIGURES

Figure 4.1	POPF Architecture	13
Figure 4.2	IReport, Compiler Settings	25
Figure 4.3	IReport, Classpath Settings	25
Figure 4.4	IReport, Master Report Sample	26
Figure 4.5	IReport, Master Report, Fields	26
Figure 4.6	IReport, Master Report, Parameters	27
Figure 4.7	IReport, Master Report, Adding Subreport	27
Figure 4.8	IReport, Master Report, Subreport Settings	28
Figure 4.9	IReport, Master Report, Common View	28
Figure 4.10	IReport, Subreport, Fields	29
Figure 4.11	IReport, Subreport, Parameters	29
Figure 4.12	IReport, Subreport, Adding Subreport	30
Figure 4.13	IReport, Subreport, Subreport Settings	30
Figure 4.14	IReport, Subreport, Common View	30
Figure 4.15	IReport, Subreport's Subreport, Fields	31
Figure 4.16	IReport, Subreport's Subreport, Parameters	31
Figure 5.1	Organization and Authorization System, ER Diagram	33
Figure 5.2	Organization and Authorization System, Login	34
Figure 5.3	Organization and Authorization System, Services Info	34
Figure 5.4	Organization and Authorization System, Reports Info	35
Figure 5.5	Organization and Authorization System, Deployment	35
Figure 5.6	Organization and Authorization System, Unit Type Definition	35
Figure 5.7	Organization and Authorization System, Unit Definition	36

Figure 5.8 Organization and Authorization System, Mission Type Definition	36
Figure 5.9 Organization and Authorization System, Menu Definition	36
Figure 5.10 Organization and Authorization System, Role Definition	37
Figure 5.11 Organization and Authorization System, Add Menu to Role	37
Figure 5.12 Organization and Authorization System, Add Service to Role	38
Figure 5.13 Organization and Authorization System, Add Report to Role	38
Figure 5.14 Organization and Authorization System, User Definition	39
Figure 5.15 Organization and Authorization System, User Search	39
Figure 5.16 Organization and Authorization System, User Info Listing	40
Figure 5.17 Organization and Authorization System, Add Role to User	40

LIST OF ABBREVIATIONS

DVO

Data Value Object

EJB

Enterprise Java Beans

EJB-QL

EJB Query Language

ER

Entity Relationship

ERP

Enterprise Resource Planning

GUI

Graphical User Interface

HQL

Hibernate Query Language

IDE

Integrated Development Environment

JAXB

Java Architecture for XML Binding

JBPM

Jboss Business Process Management

JDO

Java Data Object

JPA

Java Persistence API

JSF

Java Server Faces

MVC

Model View Controller

ORM

Object Relational Mapping

O2R

Object to Relational

PDF

Portable Document Format

POJO

Plain Old Java Object

POPF

Productivity Oriented Programming
Framework

RDBMS

Relational Database Management
System

SOA

Service Oriented Architecture

SQL

Structured Query Language

XML

Extensible Markup Language

CHAPTER 1

INTRODUCTION

Great shortage of software industry is the difficulty of the updates of the technologies invested. The only money losing issue that is planned by the investors of a product -for what money and time is spent, staff are trained, coding and testing is done, and also is being operated- is the maintenance of the product to satisfy the customers. However, the technology usually improves faster than companies' product development speed. It is a disaster to leave the product in cold and start all over from the beginning with the knowledge that is obtained throughout the past development experiences. Today even large companies are settling with their old applications which are tested and proven. To be updated without interruption is a big trouble for a running application. This process must be performed by a soft transition for the continuity of success in software.

Most of the professional software companies have embarked preparation a framework to transfer their experience from past projects to future ones. They have built flexible structures for the solutions to common problems that can be faced in all projects and created a framework by integrating them. It can be said that in most cases this preparation is useful. It is also clear that there are many examples which show great success in this study.

But over time, created frameworks dropped behind the technology, applications using these frameworks have not satisfied customer demands. The customer faced with software manufacturer with a new technology request and has continued to request a new one although current one satisfies the needs. During this time, innovative solutions of newly established firms has strained present products with their new technology advantage. Competitive market started to weaken old products although they were proven.

Obvious thing to do each company, whose target is to earn money and sustainability, is to constantly stay up-to-date. In doing so, to ensure re-use of previous work as much as possible is an important factor to reduce cost and time. So, frameworks should have a structure that allows them to be shaped according to the technological developments.

Being in a modular structure for prepared products is an important factor for reusability. The software components which are doing a specified job and can be seen as a black box can reduce software cost in most projects they joined by doing their job perfectly. However, the package based on a certain infrastructure, only can be used in projects that use the same infrastructure. To overcome this problem, the software brings a different perspective and a new approach called SOA which addresses the operations -that cannot be logically apart- as services [1] has been developed. In this way, software services which have been developed by using different technologies, create a protocol that can be counted as universal and communicate over this protocol. Although this approach is the same as traditional programming methods, it comes forward with the differences brought to the software development approach. This issue will be addressed in Chapter 4.

In this study, especially the E-Government and ERP applications are considered. Because it can be easily seen that these kinds of applications are the largest and comprehensive projects of the IT sector. Although these projects have a very low complexity and the only works they have done are create-read-update-delete operations, they either get the largest pay of the pie or face a great failure. The reason why this happens should be examined.[2]

With regard to observations, it can be said that ensuring productivity is the basic problem. The essential point of this discourse is that huge projects -which are managed by experienced staff according to various quality standards-, can result in failure. Besides, by giving the look at the construction sector, it can be seen that a non-contractor who has not been educated about techniques is able to draw up a project by the help of a civil engineer and an architect and build a big building on time with the equivalent budget of a software project. For a contractor who has constructed same kind of buildings, it can be said that his work is routine and he is using an already succeeded design draft. But it is also important to note that the software industry has a history for almost fifty years and there are many success stories. In fact, sharing information and sharing of success stories seem to be much more effective in the software industry. Then again, the reasons that affect the linear success should be re-examined.

Human factor is the clearest answer to give. The most important factor in productivity which causes an obstacle is psychological status of developers. Routine coding developers can focus to work in a format up to 4 hours a day. For the managers who try to repeat their own success examples in the past, the door to success is the repetition of the behaviors of their past employees. The performance of people, who are using framework to develop the application, is seen as the necessary work to ensure the emergence of the product. This approach is true in some cases because the facts of life are engaged. In a way, the gains are wanted to be reused. But this means copying the past defects. A simple error of abstraction, a simple dependence, an omission of a control that can be addressed at a common point or a wrong established infrastructure approach come back as rules that the programmers have to obey and unnecessary complexity. Moreover, such solutions do not fit into any generally accepted technology; they raise also the obligation for the programmers to maintain the infrastructure. As a result, army of programmers who work overtime and day and night without complaining has been trying to overcome the same problems. It is being too late even this problem has been wanted to be solved. A functional product is needed to be seen by the customers. Bad code sets in, is written in hurry and the final product emerges at the end of overtime work. The following day, another so called rule sets in: Working code should not be touched![3]

Cost calculation is essential for businesses. Of course, a product which cannot be sold will not bring any return but let us assume a company which has good marketing skills, and a specific position in the market. For the companies that have been doing business in crisis environments, it is important to decrease cost and to know that every penny spent would correspond at the development of the product. The main factor that increases the cost is to resolve the same errors again and again and have non-reusable code by writing bad code. Productivity increase is vital to keep the business standing. Technologically being up-to-date has an important place in the market share protection.

The difference of this study compared to the others is that, it is prepared by taking into consideration all the problems described. It can be thought that the competitors have more effective solutions to specific problems. However, the benefits will come out when POPF is not considered just as a software infrastructure and its affect to the total benefit cost is taken into count.

CHAPTER 2

PRODUCTIVITY IN SOFTWARE

In order to be able to talk about productivity based on government applications and ERP based software applications, the factors affecting the productivity should be determined first. The sufficiency of the environment, in which the product is developed, comes into prominence when using such programming methods in development of these kinds of products. Programmers are expected to find solution to all problems that pops up during the development phase in such inconvenient environments, which in the later case leads to taking action without doing any sufficient pre-study. This situation causes common problems to be handled repetitively. As result, the quality of products lowers and maintenance gets more difficult. Moreover, products cannot reach a particular level of quality standard.

The following sections mention problems affecting the productivity in software:

2.1 Persistency

Stableness, which is a certain need for any application, should be handled with generally accepted and portable methods. Even though DBMSs constitute the most common solution for saving different kinds of data, it should not be forgotten that it is not the only solution. However, depending on the selected persistency for the product, the base used for saving the data should be planned very well for operations like creating, reading, updating and deleting related information from the system.

If it is considered that the persistency environment is a database, the single entity operations should certainly be feasible and software modules constituting the product should be designed in a way so that they are not affected by a possible future change to be done in the access to

database or database server.

The method used in accessing to database should be defined clearly and supported by APIs that are easy to use and effective.

2.2 Reporting

Reporting is generally handled after completion of the development phase. If the reporting is planned in a well organized way, it eases the work considerably during the development phase too.

It has been observed that many programmers create their own queries and save them in private text files. If these queries can be saved at a common place and used later when writing product report, the reporting can actually be done much more efficiently. This would considerably ease the amount of work and shorten the time spent for the reporting.

2.3 Logging

It is an obligation to have logging for observing the performance during the project development. A well written log would positively affect the quality of code written by programmers. Access to logs should be provided from all possible environments. Reporting should be able to be done with the help of logging information, e.g., owner of the log or date of the log.

2.4 Organization and Authentication

In larger projects, people working in the project have often different access rights to different application parts. Even if all data is saved in a common database, all taken steps and access to critical data should be hierarchically appropriate. In today's world, this problem has turned to a system that is defined by generally accepted rules. All authentication and authorization procedures should be designed based on organizational hierarchy. These procedures should be able to be adapted in a flexible way according to customer needs. It should be sufficient for the developer to have information about end user roles to make the product work properly after the development phase.

2.5 Workflow

Workflow is a need that is directly connected to management of the organization. This need pops up especially when a particular is to be done by more than one person. When considered work steps, this is purely a matter of time course. During this period people from both inside and outside of the organization may need to work. In some cases, some time critical processes should automatically be triggered. All these problems should not be left to the programmer; rather they should be handled under the concept of both Workflow and Organization principles. In addition to the Workflow list, an additional list that will help end users when using the product should also be prepared.

2.6 Scheduling

When considered the whole system's performance, requests that are not time critical should be postponed, all requests should be scheduled according to priority levels, all requests should be handled in time and the result should be returned to the client timely. If a system has been developed according to service based architecture, it would not be possible to handle the problem within the project itself. A scheduler that is responsible for observing the system and storing all requests that will later be executed in the system should certainly be planned in the system.

2.7 Date, Time and Exchange Rate Operations

When considering official processes, it is not that easy to realize all date, time and exchange rate calculations using a simple logic. Depending on the country or region where the system will run, working hours and holidays sometimes differ. These differences should be handled in the system and when doing calculations the system should be able to ask itself some smart questions, i.e., the number of working days between two dates, the difference in the exchange rates for a particular currency from a particular date till today, etc.

2.8 Multilanguage Support

Global market companies are international in general. There are many examples that a particular implementation is used in many countries. It should be enabled for end users from different parts of the world to be able to use the system in a particular language. Adding a new language support to the implementation should be able to be done without changing the structure of the software product.

2.9 Cache System

Data that is often read but changed quite seldom should be saved in a cache that is convenient for parallelism.

2.10 File Server

If a file is too large to save it in the database, it can then be saved in a file. For this purpose, a file server to supply fast and parallel access should be prepared in the system. A system supplying the possibility for indexing and file access authorization with fast access would be helpful for many applications.

2.11 Third Party Application Integration

Today, there are not many applications running standalone. Many applications communicate with some others. The base for being able to supply this communication and use other libraries in this process should be established. The necessary plan should be ready to establish a new communication protocol in case it is needed.

Most common problems a programmer can come across in his daily work have been listed above. There are, for sure, many other reasons affecting the productivity besides programming related ones. However, the reasons mentioned above constitute the ones that would certainly help to improve the productivity. If these problems are solved in the software development phase, it would be easier to focus on real design problems instead of losing time and energy

because of unexpected problems during the implementation of the product. If the programmer can work effectively, successful results that he will get would make a certain positive effect in his psychological health and make him much more productive.

Persistency, Reporting, Organization and Authentication, Multilanguage Support and Third Party Application Integration requirements are implemented for reference implementation.

CHAPTER 3

CURRENT STUDIES USING JAVA ANNOTATIONS TO INCREASE PRODUCTIVITY

Java annotations are simple specifications that do not have to comply with any pre-conditions. Annotations have been used with Java 5.0 version and have rapidly come to a strong position in the Java world. They have taken the place of config files through the advantage of their simple structure and ease of implementation. They have increased the tracibility by the 'In-line Definition' feature. The fact - that powerful software frameworks, such as Hibernate, Spring and Seam, becoming aware of annotations and giving support for that structure - has an important place in the undeterred rise of annotations.

These annotations providing convenience for application developers, are also very important for POPF. Therefore, the frameworks which goals productivity, uses Java annotations and can be considered as competitors of POPF will be examined.

3.1 jpa2web

jpa2web[4][5] is a code generation application that builds a complete project by using the annotations which JPA uses to define the relations between entities. jpa2web generates the codes for web based pages on which create, read, update, delete operations are performed for all the entities defined with annotations. These generated web based application can only be used for data entries. But also, this application is draft for the project that the business logic will be put on. But after the intervations done to generated code, regeneration can be needed. So all the changes done to the project will be lost. This is why this framework cannot lead to successful projects in real life. But besides these, this framework is a success in theory for

being a proof that shows how annotations turns simple pojos to strong components.

3.2 OpenXava

The OpenXava[6] framework is in effort to realize MVC pattern using POJOs that use JPA annotations. OpenXava has relatively more mature infrastructure than jpa2web. Instead of code generation approach, it has chosen to use code patterns that will work at runtime. The framework has been made configurable with XML-based configuration files. It is not suitable for very large projects. It is hard to shape the visual items. Performance seem to be not enough. It is also not suitable to develop applications by modules. However, it is a continuing project that may have future outputs.

3.3 Seahorse

Seahorse[7][8] is an annotation driven framework that aims to increase productivity and simplify programming in Java. This framework is also in effort to realize MVC pattern. It is similar to Spring Framework. The actual work done by this framework is at the model layer. Applications can be developed module by module. But this a new framework which has lots of issues to solve. There is not a feature to support authentication system. It has communication problems with external systems because it is not a service based framework. This framework can be suitable only for medium scaled projects. It is a project that has future.

3.4 Stripes

The Stripes[9] framework also uses Java annotations and targets the presentation layer of MVC pattern. It has action handlers to handle incoming requests from presentation layer. There are predefined structures to examine incoming requests' validity. It's easy to use with provided taglib. It does not have an integrated authorization system yet, but it will not be hard to add this system. It has a similar infrastructure with service-based architecture.

3.5 Spring Framework

The keywords that describes Spring[10] are advanced, high performance, designed with flexible architecture, service-based and the industry standard Java application framework. The success of this framework is proven with its capabilities. It is being used successfully in many enterprise-scale projects. However, it has strict rules to identify services. It can be integrated with many security infrastructure with no trouble. It has been affected from Aspect Oriented Programming on its architecture. This feature provides flexibility in many issues, but also restricts the traceability. Nevertheless, it's the most ambitious framework between the service-based application frameworks.

3.6 JBoss Seam

It is an integration application with the slogan 'seamless'. Its feature called 'Configuration By Exception' provides convenience for ordinary applications. It fulfills all kind of requirements for the Java World. All required arrangements are done for Spring, JSF, EJB 3.0, JPA, Java Annotations, Web Beans, JBPM, Drools, TestNG, etc... frameworks to be able to work together. JBoss Seam[11] is the largest competitor with all these features to Microsoft's .NET platform. However, the high hardware requirements is the greatest problem. It has almost everything for easy and fast application development.

CHAPTER 4

ARCHITECTURE AND CORE LAYER OF POPF

Productivity is a concept that is concerned with software development environment attributes, software system product attributes and project staff attributes[12]. POPF tries to reach the target product in the shortest time and with the minimum cost; abstracts developers' work at the highest layer; reduces technical problems; eases to focus on the parts where domain knowledge is required. High-quality software is the absolute product for POPF.

POPF is a framework that simplifies the work of developers who are untrained in terms of technology but well informed about domain knowledge and also is a framework that aims the advanced programming features to be used as standard by any developer.

Basically this framework is independent from the requirement analysis and design stages of software development lifecycle. So application developers have waited to learn about business rules, make general and detailed design for application and specify the user interfaces before using POPF. Then, at the development phase POPF which is a framework that allows reusability and fast development comes into prominence.

A structure which can expand unlimitedly has been designed in the light of the principles that have been specified to set up the framework which targets productivity. It's flexible, because it does not depend on any product or technologies except Java. It is easy to scale and is focused on manageable and modular software development. It has an approach that handles software as services in terms of providing reusability. However, all these features are integrated with the effort of trying not to recreate the wheel. POPF does not target to specify new approaches to problems which require expertise. On the contrary, it targets to integrate best solutions of these problems. By the means of these features, software can be kept consistently up-to-date and it will be cheaper to maintain. It is simple and does not contain unnecessary detail.

But, any detail can be added easily if required. It is in an effort to associate developers to its approach even if they don't know a clue about SOA. It plays a role as integrator for software applications designed using different technologies with its encapsulation feature.

To be able to include the mentioned features, some features of software infrastructures have been sacrificed.

For example, POPF accepts that it is impossible to satisfy customers' visual requirements. Because, requests of end users are endless at the GUI layer of software and to meet these needs are mandatory. In contrary to the GUI part, exaggerated requests are not usually come across at business-specific code part (business rules). However critical sections which can cause bottleneck are usually in this part. Therefore, any definitions are not done for GUI layer. This also means that regardless of client technology, the technology and the design of the server-side services do not change. It can work in collaboration with the services of different platforms by preparing an adapter layer among. Thereby, it is possible for any application to be accessed by both mobile devices and network based clients without making any change at service layer.

After POPF application developers make their database and user interface design, they can call the services of their modules by using the communication infrastructure provided by POPF Server (Business Server).

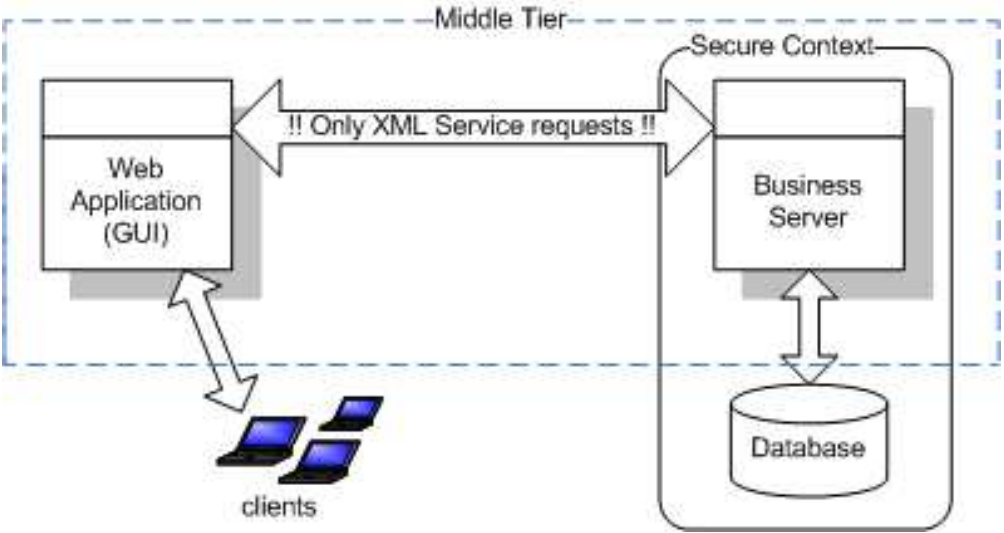


Figure 4.1: POPF Architecture

4.1 POPF Implementation : Business Server

A variety of Java technology and design approaches are used while implementing Business Server. In this part of the thesis, all steps from the server startup to message transmission between client and server are examined.

For the persistence layer of the system JPA [13] is preferred. JPA is a specification that stands over ORM tools. It is a free API that exists in Java. Actually JPA does not perform any O2R mapping process. JPA is a specification that is used to access ORM tools like Hibernate [14] or Toplink [15]. All kinds of database operations can be performed at the time of coding with this API. At runtime, the prepared configuration files and the libraries of the actual ORM tool are wrapped together to achieve communication with the database.

The powerful and comprehensive API and making applications independent from ORM tools are among the advantages of JPA. There are many ORM tools such as Hibernate, Toplink and JDO [16] which show different performances on different platforms. To have the chance to make a choice between these tools according to platform that our product will run on provides a big advantage at the competition with other products.

MySQL [17] RDBMS is used as the database product. This product is chosen because it is a free product and has high performance. Also correctness of this choice is proven with the ease of access of the documents over the Internet and the popularity of this product on forums.

Since MySQL is selected as database, Hibernate which is also free and known as a successful coworker of MySQL is selected as O2R mapping tool.

Messages between server and client are done by xml formatted texts. Xml is a file format which is easy to parse and is supported by all technological platforms.

The message is a Java object inside the server. The client creates the message as a message object and this object is converted to xml before sending it to server. When the server receives the message, the xml message is converted to the object again. JAXB [18] API is used to perform these object to xml conversions in a fast and effective way. The JAXB API is a specification just like JPA so Apache JaxMe [19] library is used as the implementation of this API.

During the development of the service infrastructure of the system, the definition of methods that represents the services is done by an unusual method - byte code engineering- instead of the Java Reflection API which is commonly used to solve this problem. This method is used to provide speed and flexibility while defining and calling services. After search, it is seen that the best library that implements this API is JavaAssist [20]. Another library called Scannotation [21] and JavaAssist is used together and the services are defined as they are classes of the system. Some features were added to these defined services to eliminate some requirements like service searching.

While designing the system, it is aimed to cover the reporting needs as a basic feature of the system. Therefore, Jasper Reports[22] which is a proven reporting tools that is able satisfy the needs of large-scale projects is embedded to system core. The data sources of the reports can be prepared as services. So, software developers can design generic services that can be used by both reports and user interfaces. This method reduces the amount of work that is done by developers.

When all these choices come together, the application developers are only responsible for developing their services and reports in a regular way.

4.1.1 Service Production

1. Definitions

Any sort of atomic operations defined on Business Server is handled as a service definition. At this part of the documentation, the rules that a developer should follow while defining a service will be explained.

To define a service, a regular POJO is defined and this POJO has methods with the signature

```
public static Message xxxxxx(Message message Context context)
    throws Exception
```

and the annotation '@Service'. There is no need to make any other identification.

```
@Target(ElementType.METHOD)
@Retention(RetentionPolicy.RUNTIME)
```

```

public @interface Service
{
    String name();
    String desc() default "";
    Param[] inputParams() default {};
    Param[] outputParams() default {};
}

```

If examined, it will be seen that the only mandatory attribute is 'name'. That's because, it is possible for a service not to have any input or output parameters. On the side, although 'desc' (description) attribute is not mandatory, it's recommended to fill that field because this description is viewed at developer support web site. In the picture below, the details of '@Param' annotation are given:

```

@Target(ElementType.METHOD)
@Retention(RetentionPolicy.RUNTIME)
public @interface Param
{
    String name();
    Class type();
    String desc() default "";
    boolean canBeNull() default false;
    boolean canBeEmpty() default false;
    String comboConstantName() default "";
    ParamDetail[] details() default {};
}

```

It's shown that a parameter should have 'name' and 'type' attributes but the 'desc', 'canBeNull' and 'canBeEmpty' attributes are left optional.

At Service Oriented Architecture, the end user parameters of service requests are not trustworthy. Therefore, it's an obligation to check every input parameter in the service code. This requirement means that same logical control codes will be used repetitively.

An attribute is added to service definitions to overcome this issue. 'canBeNull' attribute defines if the value can be sent to server as null or not. When the service request comes, an automatic control is done according to service definition. In case of noncompliance,

related error messages are returned.

A 'false' value in 'canBeNull' attribute means that the parameter should always have a value. But in some cases it comes to mind that there can be parameters which are logically blank but can not have null values.

A 'false' value in 'canBeEmpty' attributes means that the parameter can neither have a null value nor be blank. The meaning of this control differs from one data type to other. For example, for String data type the empty value expressed by "" and for array-type variable, zero sized array means blank.

'canBeNull' and 'canBeEmpty' attributes are false by default. So, all parameters must have a value as long as the direct opposite is indicated.

As an example, a simple service is examined:

```
@Service
(
    name="ORGANIZASYONDA_MENU_ELEMANI_GUNCELLE",
    desc="Menü hiyerarşisindeki bir menü adımlarını günceller",
    inputParams=
    {
        @Param(name="menuItemId", type=Long.class),
        @Param(name="ataMenuItemId", type=Long.class),
        @Param(name="menuIpucu", type=String.class,
            canBeNull=true, canBeEmpty=true),
    }
)
public static Message organizasyondaMenuElemaniGuncelle
    (Message message, Context context) throws Exception
{
    Long menuItemId = message.getLongParam("menuItemId");
    Long ataMenuItemId = message.getLongParam("ataMenuItemId");
    ...
}
```

The 'Message' object wraps requests from outside the world. The most important variables of Message class are specified below:

```
@XmlElement
@XmlAccessorType(XmlAccessType.FIELD)
```

```

public class Message
{
    public enum MESSAGE_TYPE {SERVICE, REPORT};
    public enum STATUS {SUCCESS, ERROR, WARNING, SYNC_ERROR};
    public enum REPORT_OUTPUT_TYPE {JRPRINT, PDF, WORD, EXCEL, TXT};
    private MESSAGE_TYPE messageType;
    private STATUS status;
    private String statusMessage;
    private REPORT_OUTPUT_TYPE reportOutputType;
    private byte[] reportContent;
    private String moduleName;
    private String requestName;
    private SessionData sessionData;
    ...
}

```

There are two types of messages: 'service' or 'report'. The reporting system is designed at the core layer with the same architecture of services that's why 'report' is one of message types. Through this way, a service providing data to client-side interface is also able to be used to provide data to a report.

In the case of message type being 'report', the output file format of the report should be specified. This specification is done by the help of a parameter named 'REPORT OUTPUT TYPE' whose type is PDF by default. Other possible output formats are Word, Excel, TXT and JRPrint. JRPrint which is the native output format of Jasper Reports is supported to make possible if the client needs Jasper Reports Embedded Viewer (JRViewer) in his application. As understood JRViewer can only work with Jasper's native format.

The 'status' attribute is meaningful in case of message returns. If status is 'ERROR', there happened an error on the server side during the service execution and the transaction started for this service is rolled back. But if the status is 'WARNING' or 'SUCCESS', this means the transaction is successfully completed and a commit operation is performed. For all cases, the related message of the status is sent within the attribute 'statusMessage'.

In traditional service name mapping, methods are called by appending module names

in front of service names. But this method makes the structure of name switching mechanisms complex and slows down the service method search. To overcome this problem, a different way is followed at POPF. Each service is kept in relation with the java archive file (JAR) that it is deployed in. A module definition is done with one jar file. All services under this module are meaningful with the name of this module. The reports of the module are defined in the same way. The reason of this approach is to enlarge the namespace and reduce name conflicts.

In 'sessionData' attribute, language choice and access rights of a client are specified. Business Server assumes the web applications will build their own session management. By this way, all requests come to Business Server are from secure context. The only problem of this case is the need of control if the client has access right to reach the resource. The 'sessionData' attribute is essential to do these authorization controls and keep access logs reports.

2. Persistence

For each module, database definitions must be done by a configuration file named META-INF/persistence.xml. In this file the entities which were once called as DVO are defined. Each entity corresponds for a database table. In fact an entity is a simple POJO. A simple persistence.xml file looks like:

```
<?xml version="1.0" encoding="UTF-8"?>
<persistence version="1.0"
xmlns="http://java.sun.com/xml/ns/persistence"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:schemaLocation="http://java.sun.com/xml/ns/persistence
http://java.sun.com/xml/ns/persistence/persistence_1_0.xsd">
    <persistence-unit name="Logger"
        transaction-type="RESOURCE_LOCAL">
        <class>tr.com.bilisim.sample.entity.Employee</class>
        <class>tr.com.bilisim.sample.entity.Department</class>
    </persistence-unit>
</persistence>
```

Stored procedures or native SQL sentences are not used at the preparation of services. Rather than this, the EJB-QL [23] which has a similar syntax with HQL [24] is used by

JPA.

An Interface named EntityManager is used to find, add, update and delete the Entity classes. Business Server handles a transaction per request. But this does not mean the transactions are service based. Because one request can contain more than one service. EntityManager interface consists of required transaction handling methods. Developers don't need to think about the transaction management. But the system is designed to make possible for the modules to be deployed separately. This feature will be useful for distributed deployment strategies. If service request has different services which are deployed in different modules, one transaction is created per each module change.

EntityManager access is done via 'context' object. Client message is automatically sent to service with context object. An example of this is as follows:

```
Query q = em.createNamedQuery("OrgRol2Kaynak.findByCriteria");
q.setParameter("TIP", tip);
q.setParameter("ROL_ID", rolId);
q.setParameter("DEGER", deger);

try
{
    return (OrgRol2Kaynak)q.getSingleResult();
}
catch(NoResultException e)
{
    ...
}
```

All entity objects are marked with @Entity annotation. @Service and @Report annotations are special to POPF. However, @Entity is a JPA's annotation.

```
@Entity(name = "ORG_ROL")
public class OrgRol implements Serializable
{
    @TableGenerator(name="PK_ORG_ROL", table="PK_GENERATOR",
                    allocationSize=1)
    @GeneratedValue(strategy=GenerationType.TABLE,
                    generator = "PK_ORG_ROL")
    @Id
```

```

@Column(name = "ID", nullable = false)
private Long Id;
@Column(name = "AD", nullable = false)
private String ad;
@Column(name = "ACIKLAMA")
private String aciklama;

@ManyToMany(cascade = CascadeType.PERSIST)
@JoinTable(name = "ORG_ROL_2_ROL",
           joinColumns = {@JoinColumn(name = "ANA_ROL_FK")},
           inverseJoinColumns = {@JoinColumn(name = "ALT_ROL_FK")})
private List<OrgRol> roller = new ArrayList<OrgRol>();

public OrgRol(){
public Long getId(){ return Id; }
public void setId(Long id){ Id = id; }
public String getAd(){ return ad; }
public void setAd(String ad){ this.ad = ad; }
public String getAciklama(){ return aciklama; }
public void setAciklama(String aciklama){ this.aciklama = aciklama; }
public List<OrgRol> getRoller(){ return roller; }
public void setRoller(List<OrgRol> roller){ this.roller = roller; }
}

```

4.1.2 Report Preparation

There are rules to follow during preparation of a report using Jasper Reports. These rules will be reviewed on a sample report, respectively.

Reports are not prepared via embedding SQL sentences into report templates. Report templates are prepared independent from data sources. Data sources are Java classes which are defined like services.

Writing database queries of a sub report of a master report is the starting point to create a report. After the templates and data sources of a sub report are prepared, the master report is prepared to integrate the subs.

After examining the class below, the use of @Report annotation will take attention. This annotation is used to mark the definition of the report to introduce it to the system. Report name, description, master or sub report info and template file path information are specified by this annotation. This class defining Report also has to implement the 'JRDataSource' interface. The methods of this interface are used during filling the template file with data. So, report data must be prepared at the time of class construction. The database is not the only source to produce data for report. A Service, an xml file or a web service can also be used as report data source.

```
@Report(name="MY_SUBREPORT_1",
        menuName="Benim Birinci alt-Raporum",
        desc="Deneme Raporu - MY subreport 1 açıklaması",
        masterReport=false,
        templatePath="tr/com/bilisim/isy/logger/report/subreport.jasper",
        params=
        {
            @Param(name="dept_id", type=Integer.class, desc="Departman numarası")
        })
public class MySubreport1 implements JRDataSource
{
    private List<Employee> data;
    private Iterator<Employee> iterator;
    private Employee current;

    public MySubreport1(Message message, Context context)
    {
        EntityManager entityManager = context.getEntityManager();
        String queryString="SELECT e FROM tr.com.bilisim.sample.entity.Employee"+
            " e WHERE e.dept_id = ?1";
        Query query = entityManager.createQuery(queryString);
        query.setParameter(1, message.getIntegerParam("dept_id"));
        this.data = query.getResultList();
        this.iterator = this.data.iterator();
    }

    @Override
```

```

public boolean next() throws JRException
{
    if(iterator.hasNext())
    {
        this.current = (Employee)iterator.next();
        return true;
    }
    this.current = null;
    return false;
}

@Override
public Object getFieldValue(JRField field) throws JRException
{
    if(this.current == null)
        throw new JRException("Hata! Lütfen next() metodunu kontrol ediniz !");

    if(field.getName().equals("emp_id"))
        return this.current.getEmp_id();
    else if(field.getName().equals("dept_id"))
        return this.current.getDept_id();
    else if(field.getName().equals("fname"))
        return this.current.getFname();
    else if(field.getName().equals("start_date"))
        return this.current.getStart_date();

    return null;
}
}

```

```

@Target(ElementType.TYPE)
@Retention(RetentionPolicy.RUNTIME)
public @interface Report
{
    String name();
    String desc() default "";
    String menuName() default "";
}

```

```
boolean masterReport();
String templatePath();
Param[] params() default {};
}
```

The next() method is used by Jasper Reports to check if a new row exists or not in the data source which is used during data insertion to the report template. The getFieldValue (JR-Field field) method is used to get the value of the column specified by the parameter. While implementing Data Source class, these methods must be handled by developer.

The following definitions were made to be used in the example:

```
@Report(name="MY_SUBREPORT_2",
        menuName="Benim ikinci alt-Raporum",
        masterReport=false,
        templatePath="tr/com/bilisim/isy/logger/report/subreport1.jasper")
public class MySubreport2 implements JRDataSource
```

```
@Report(name="MY_FIRST_REPORT",
        menuName="Benim İlk Raporum",
        masterReport=true,
        templatePath="tr/com/bilisim/isy/logger/report/masterreport.jasper")
public class MyFirstReport implements JRDataSource
```

The next step is the preparation of the report template using iReport tool. First of all, iReport tool is needed to be configured. These settings are done to make it possible for the compiled versions of report templates to be saved in the directory where report template exists.

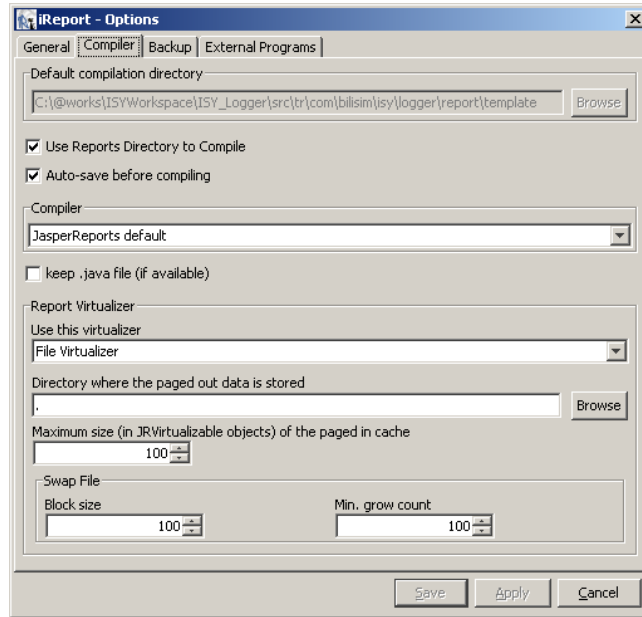


Figure 4.2: IReport, Compiler Settings

Besides these settings, two system libraries which make parameter passing to the report templates possible must be at the classpath during this operation:

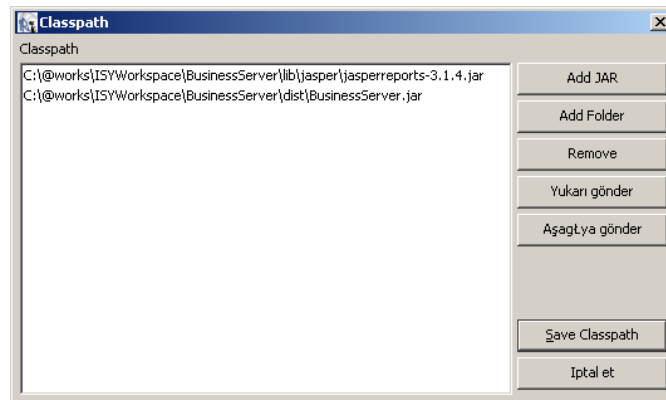


Figure 4.3: IReport, Classpath Settings

The master and the sub reports will be prepared respectively. An example master report is seen below:

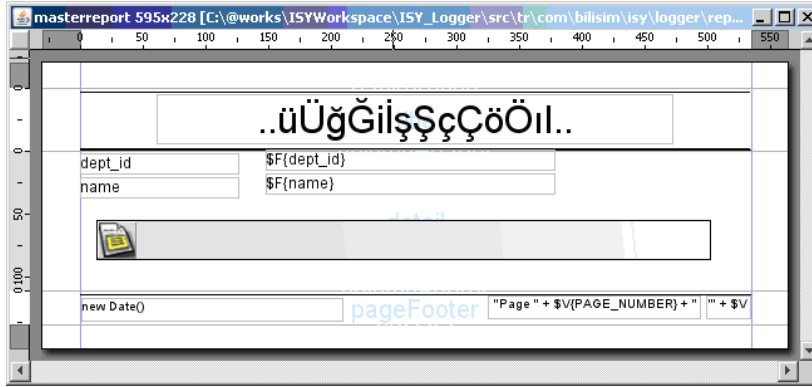


Figure 4.4: IReport, Master Report Sample

DataSource class of the sample master report supplies two columns of fake data. At the report template, the definition of these fields is done as below:

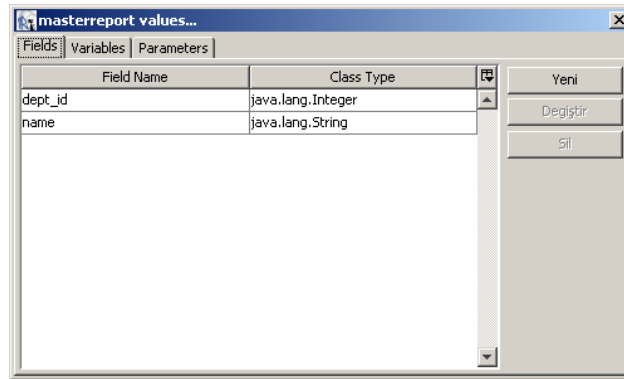


Figure 4.5: IReport, Master Report, Fields

The 'message' and 'context' parameters are passed into the constructor of DataSource class automatically. However, these parameters should be defined in the report template to be able to be used. But there is no need to define these parameters if no sub-report exists.

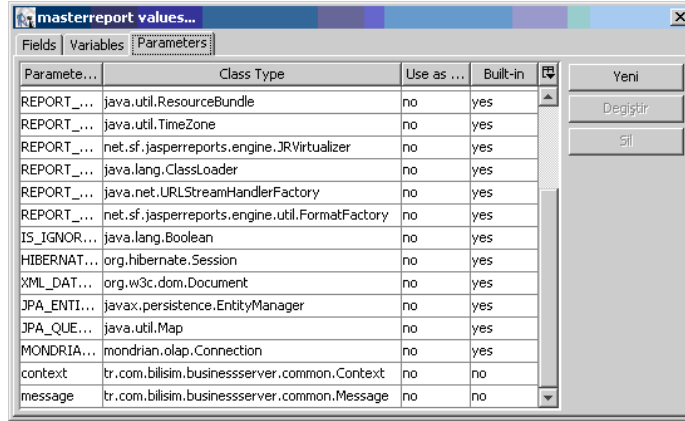


Figure 4.6: IReport, Master Report, Parameters

After right-clicking on the report template and selecting the properties menu step, the screen below appears. From this step, 'Use data source expression' step must be selected and the definitions of the sub-report and parameter must be done like image below. To be able to do all operations in one source code line, the 'add' and 'set' methods of Message class are designed to return a reference of its own.

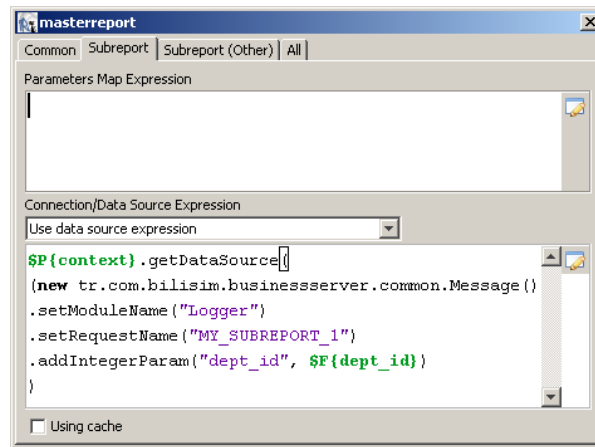


Figure 4.7: IReport, Master Report, Adding Subreport

At the next tab, the sub-report template is specified. Here, the 'Subreport Expression Class' field must be selected as 'net.sf.jasperreports.engine.JasperReport'. If a new sub-report will be defined in the current sub-report, the message and context parameters should be passed

to new sub-report in 'Sub-report parameters' section. If requested, some extra parameters can also be defined here for visual purposes. If not required, the message parameter can be undefined. But for all cases, if there is a sub-report, the context parameter is needed.

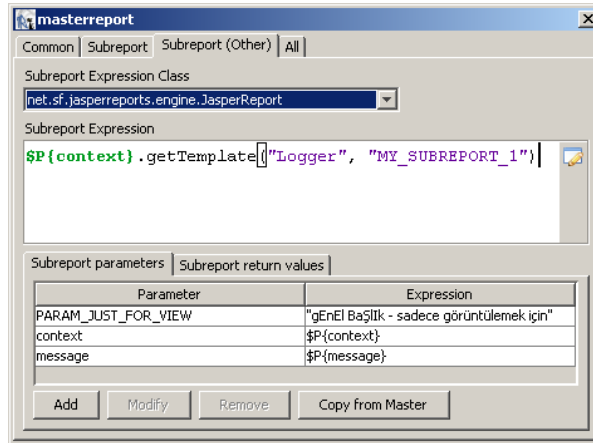


Figure 4.8: IReport, Master Report, Subreport Settings

The following image shows the first-level sub-reports:

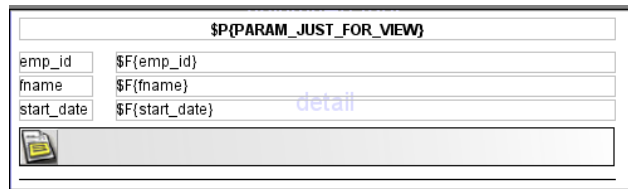


Figure 4.9: IReport, Master Report, Common View

Here, the column names which are used in DataSource class should be defined:

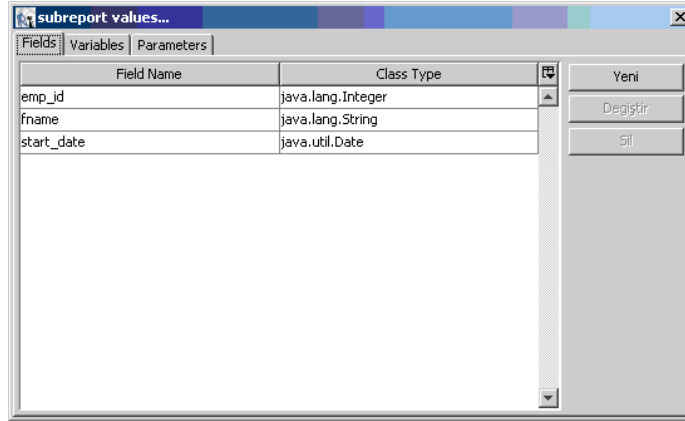


Figure 4.10: IReport, Subreport, Fields

The 'message' and 'context' parameters required to define a sub-report are as follows:

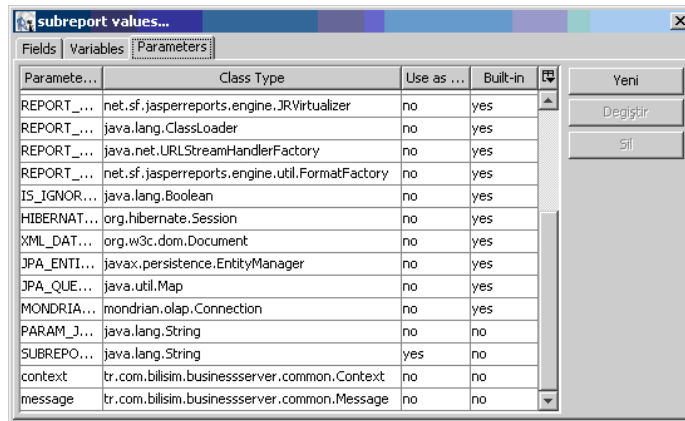


Figure 4.11: IReport, Subreport, Parameters

The definitions required to define second level sub-report on this sub-report:

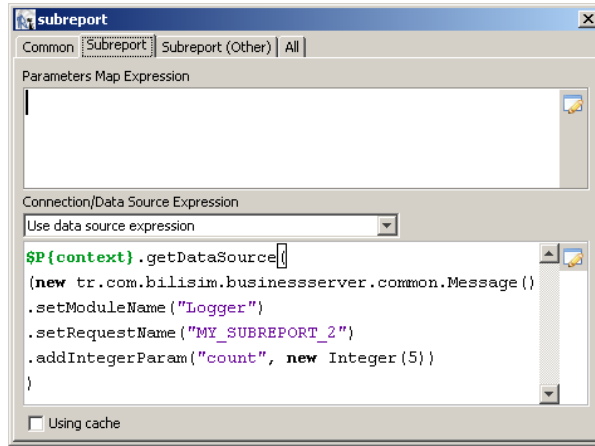


Figure 4.12: IReport, Subreport, Adding Subreport

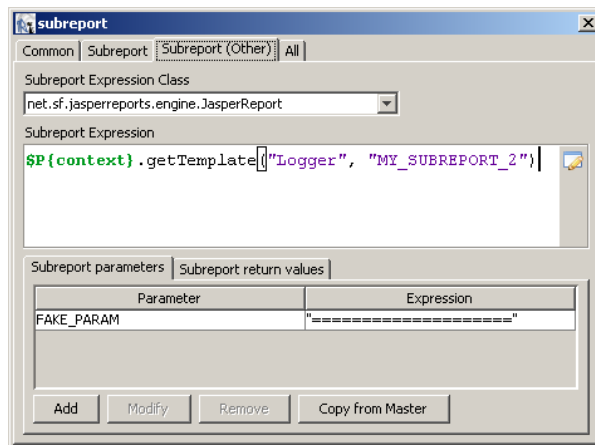


Figure 4.13: IReport, Subreport, Subreport Settings

The second level sub-report looks like as below:



Figure 4.14: IReport, Subreport, Common View

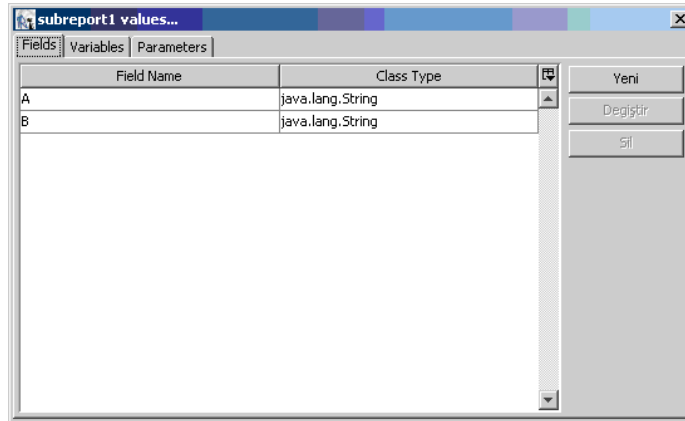


Figure 4.15: IReport, Subreport's Subreport, Fields

In this example, 'message' and 'context' parameters are not defined because there is sub-report defined. However, the input parameters from the higher level must be specified.

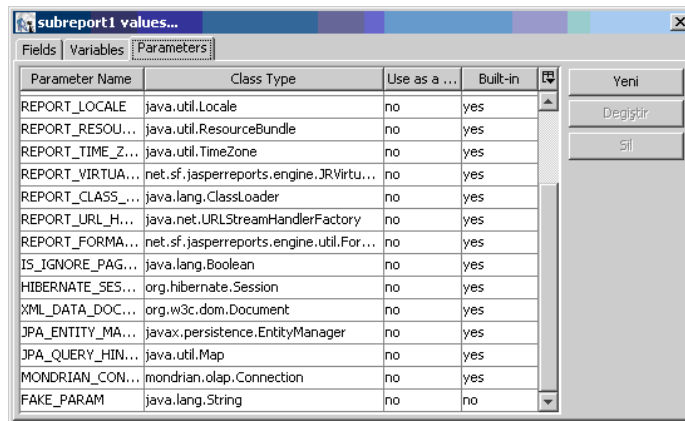


Figure 4.16: IReport, Subreport's Subreport, Parameters

So, what happens in the core layer is described. By using this core, other layers that have specialized missions to implement the POPF requirements have been added. The next chapter will examine a sample layer.

CHAPTER 5

SAMPLE REQUIREMENT LAYER FOR POPF: ORGANIZATION AND AUTHORIZATION SYSTEM

It has been decided to develop Authorization System to sample the implementation of the requirements that were set for Business Server. However, some organization definitions have been needed to be made for the authentication process to be carried out. By generalizing this need, Organization System was designed and it has been decided for this system to be integrated with Authorization System.

In POPF specification, it was specified that any description of the GUI parts would be done. However, while implementing the GUI layer, the need to prepare the menu will be formed. The menu to be formed should be prepared specific to user and this can be done only done by using authentication system. For this reason, the menu structure should be managed by Business Server like application services.

During the establishment of authorization system, the organizational structure is expected to be fully implemented according to the system working formally. However, a different way will be followed with POPF .

Users will be created without any connection to the institution units that they are working for. Then the assignment records will be created and associated with users. By this way it will be possible for users to work for more than one unit and it would become easy for users to represent their colleagues during a temporary duration.

Unit type definitions and the hierarchical structure between the units are defined. Each assignment record is composed of the related user, the related unit and the related assignment type.

The roles are designed in a way that they may contain other roles. So the hierarchical role definitions can be specified.

Each role has Resources which will be included to the authentication. These resources can be in the type of a Service, a Report or a Menu.

The roles are not assigned to users, they are meant to be assigned to assignments. Each task can have multiple roles.

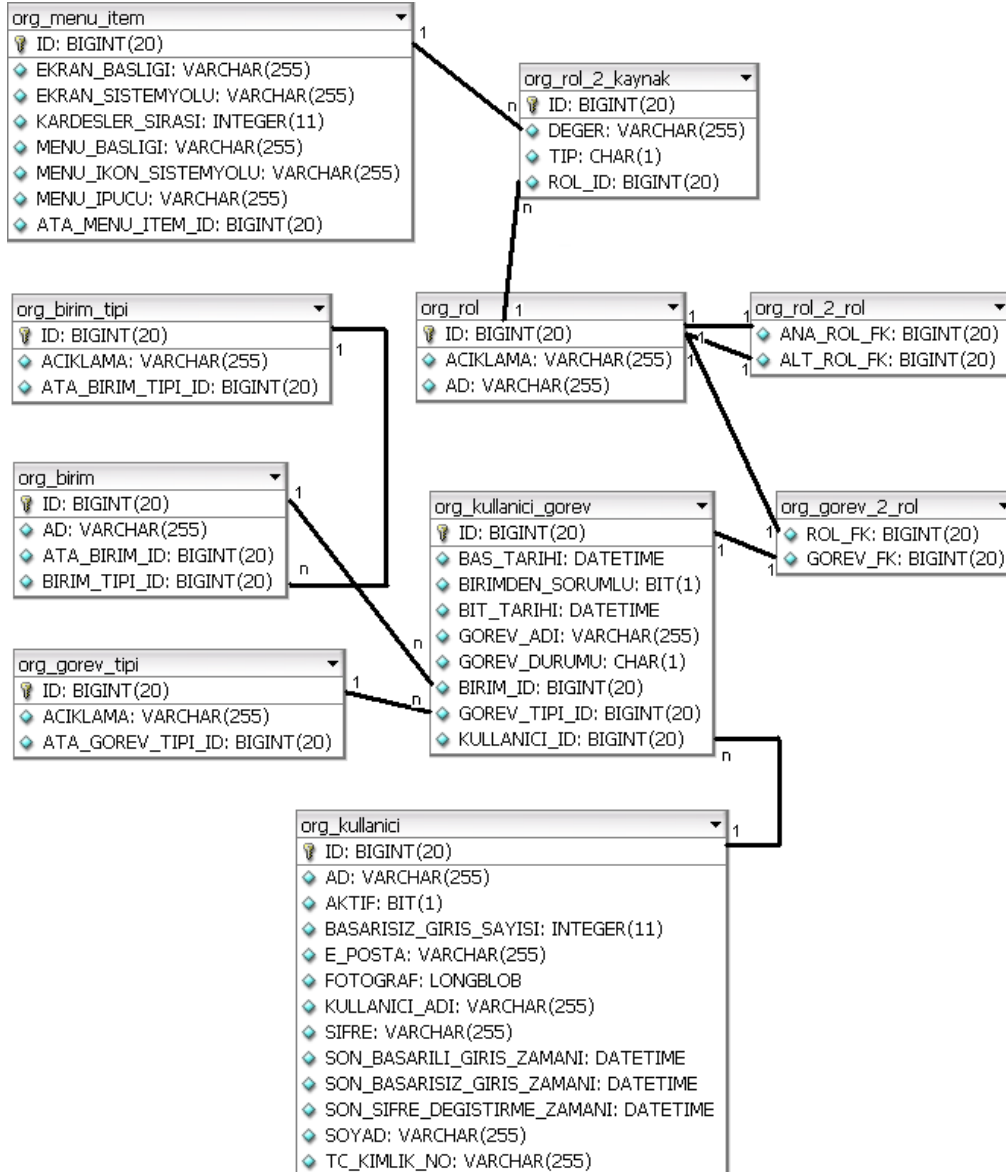


Figure 5.1: Organization and Authorization System, ER Diagram

The view of the Organization and the Authorization System which was prepared appropriate to definitions, will be examined. During this examination, the user interfaces which were prepared to manage the service definitions and the reports will be reviewed.

1. The User logs in to system with a user name and password.

İş Sunucusu - Güvenli Giriş



ÇANKAYA ÜNİVERSİTESİ

Lütfen Kullanıcı Adınızı ve Şifrenizi Giriniz

Kullanıcı Adı:

Şifre:

Figure 5.2: Organization and Authorization System, Login

2. At the menu step *Information / Services*, description and parameters (input and output) of all services are displayed grouped by modules.

İş Sunucusu Yönetimi Bilgilendirme Geliştirici Organizasyon

[Servis Bilgileri]

UYARI ! Burada yalnızca aktif olarak tanımlanmış olan modül ve servis bilgileri yer almaktadır. Yeni eklenen servislerin görüntülenmesi için sunucunun yeniden başlatılması gerekmektedir.

[Sayfayı Yenile](#)

1. Modül Seçiniz:

2. Servis Seçiniz:

Modül: Blsistem, Servis: GET_MODULE_SERVICE_PARAMETERS_INFO

Açıklama: Service's Input-Output Parameters' Info

Giriş Parametreleri				Çıkış Parametreleri		
Parametre Adı	Parametre Tipi	Par: NULL	BOŞ	Parametre Adı	Parametre Tipi	Parametre Açıklaması
moduleName	java.lang.String	false	false	desc	java.lang.String	
serviceName	java.lang.String	false	false	inputParams	java.lang.String[][]	
				outputParams	java.lang.String[][]	

POPF Gerekleştirimi | Ozan Ali Kaya | Çankaya Üniversitesi 2009

Figure 5.3: Organization and Authorization System, Services Info

3. At the menu step *Information / Reports*, description and required parameters of all reports are displayed grouped by modules.

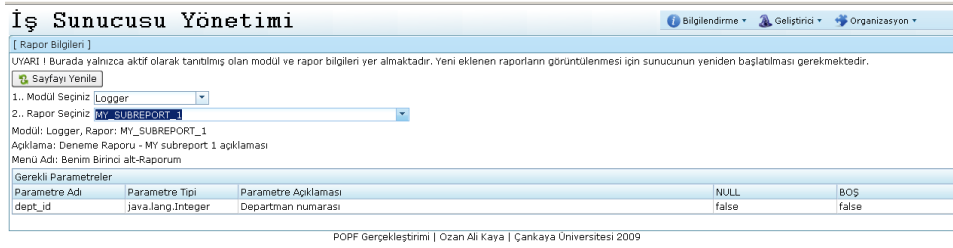


Figure 5.4: Organization and Authorization System, Reports Info

4. At the menu step *Developer / Deployment*, a new version of any library file can be uploaded to the system or any library file which is being used in the current version of system can be downloaded.



Figure 5.5: Organization and Authorization System, Deployment

5. At the menu step *Organization / Unit Type*, unit types can be managed in a hierarchical structure.

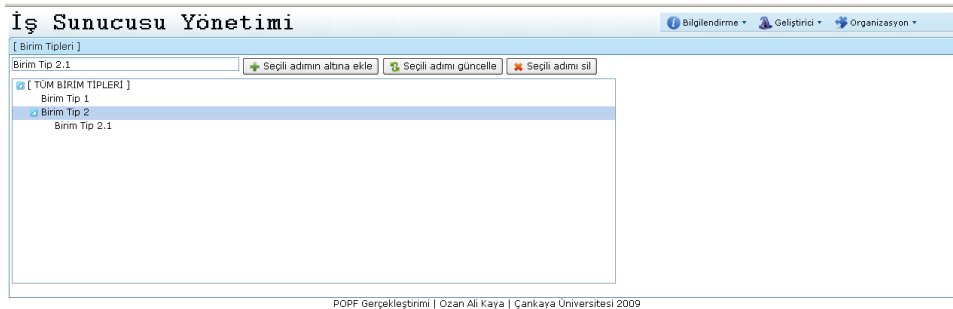


Figure 5.6: Organization and Authorization System, Unit Type Definition

6. At the menu step *Organization / Unit*, units can be managed in the hierarchy of unit types.

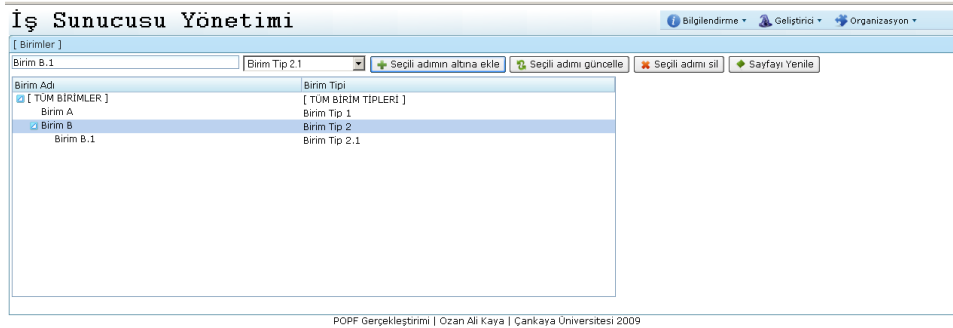


Figure 5.7: Organization and Authorization System, Unit Definition

7. At the menu step *Organization / Assignment Type*, assignment types can be managed in a hierarchical structure.

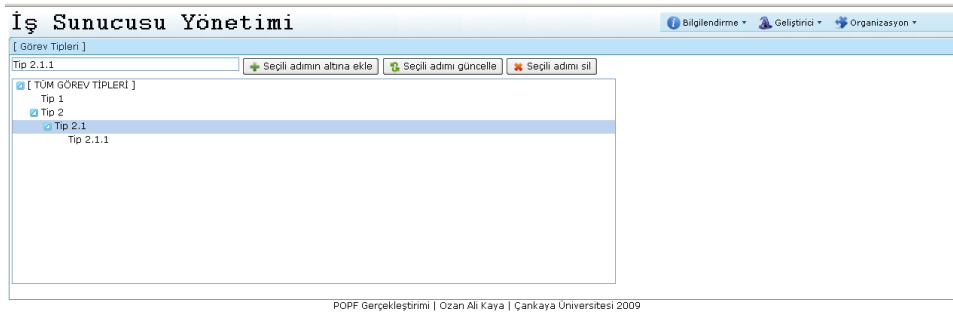


Figure 5.8: Organization and Authorization System, Mission Type Definition

8. At the menu step *Organization / Menu*, menu steps can be managed in a hierarchical structure. Menu definitions have all the details that the system which will implement the GUI layer can need.

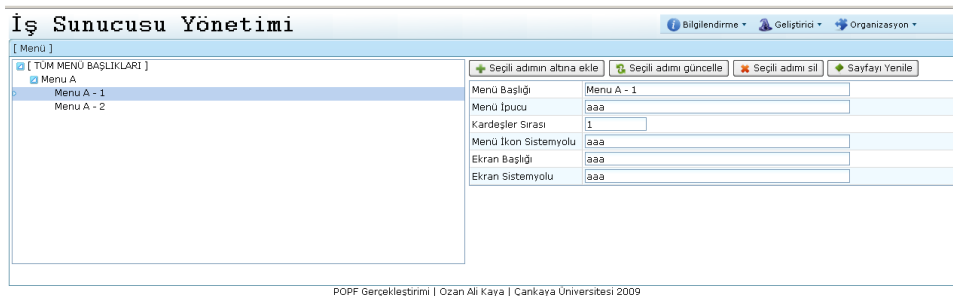


Figure 5.9: Organization and Authorization System, Menu Definition

9. At the menu step *Organization / Role*, roles can be managed. Roles can contain other roles. TO create a sub role, dragging the role and dropping it under other role is enough. Parent role is meant to include all access rights of sub-role.

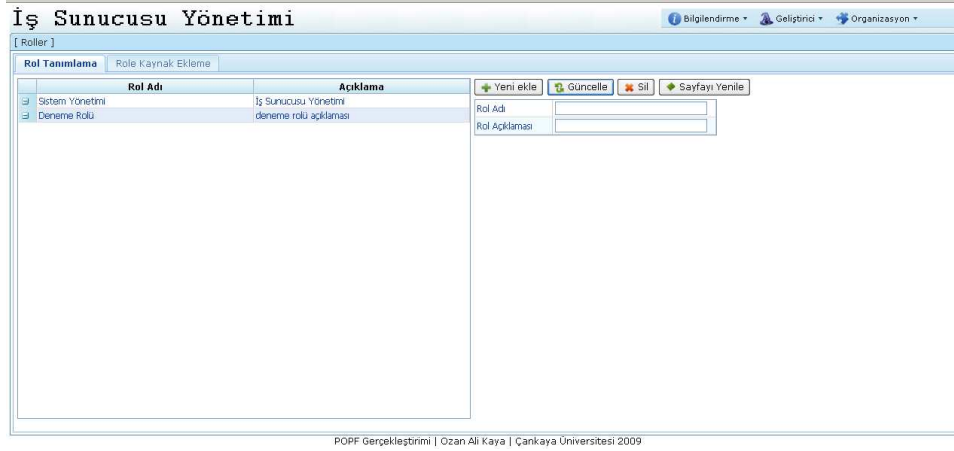


Figure 5.10: Organization and Authorization System, Role Definition

Resource can be added to the role selected. For associating a resource with a role, the resource type which can be Menu, Service or Report is marked and added to the role.

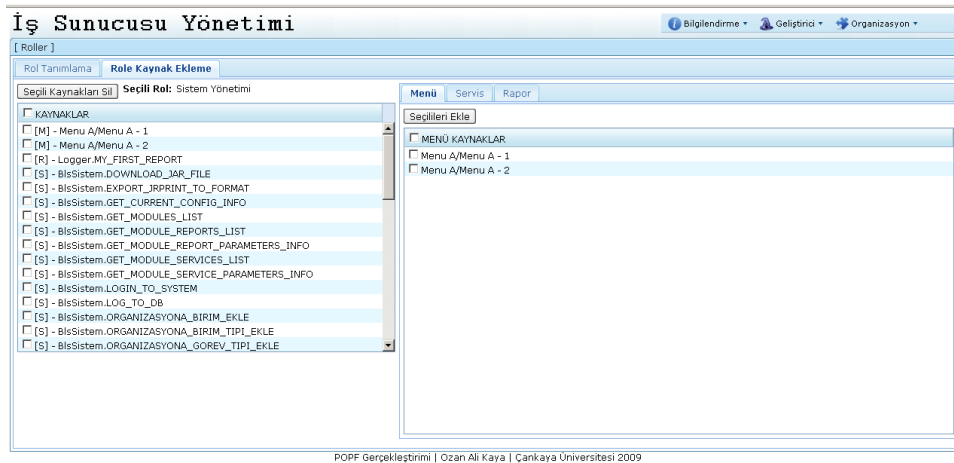


Figure 5.11: Organization and Authorization System, Add Menu to Role

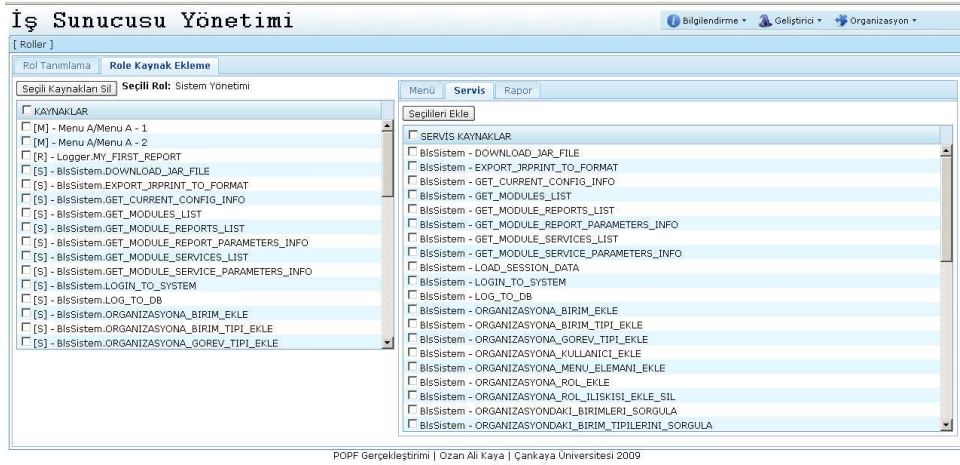


Figure 5.12: Organization and Authorization System, Add Service to Role

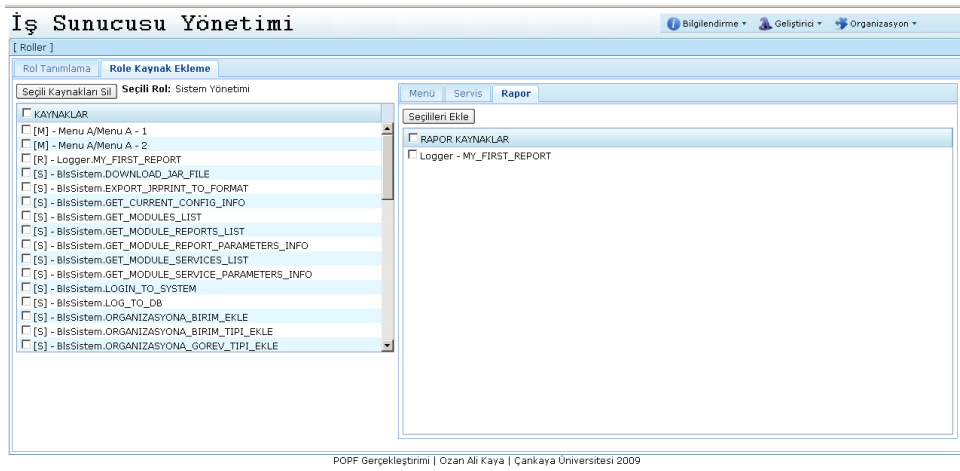


Figure 5.13: Organization and Authorization System, Add Report to Role

10. At the menu step *Organization / User* , system user definitions are managed. A new user can be defined, assignments can be added to user and roles can be added to these assignments. All user management capabilities are gathered at this menu step.

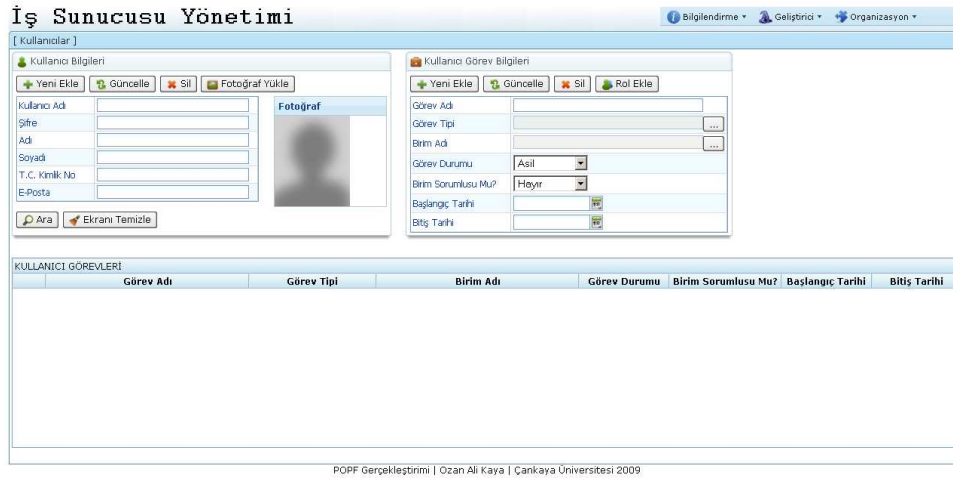


Figure 5.14: Organization and Authorization System, User Definition

In the user definition screen, the fields used to define a user are also the filter fields for search. If the user wants to perform a search, these fields should be filled according to the criteria and Search button should be pressed. Then the results will be listed as a table.

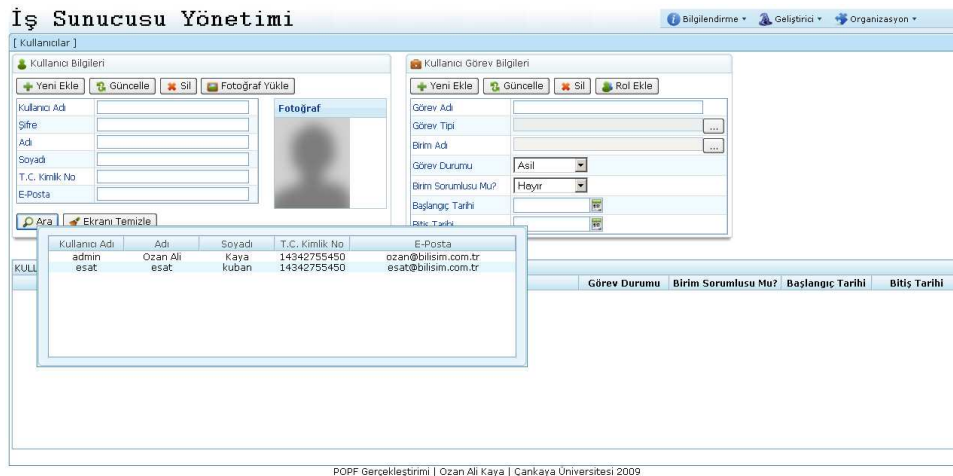


Figure 5.15: Organization and Authorization System, User Search



Figure 5.16: Organization and Authorization System, User Info Listing

After selecting any assignment of the user, Role Creation menu becomes active.

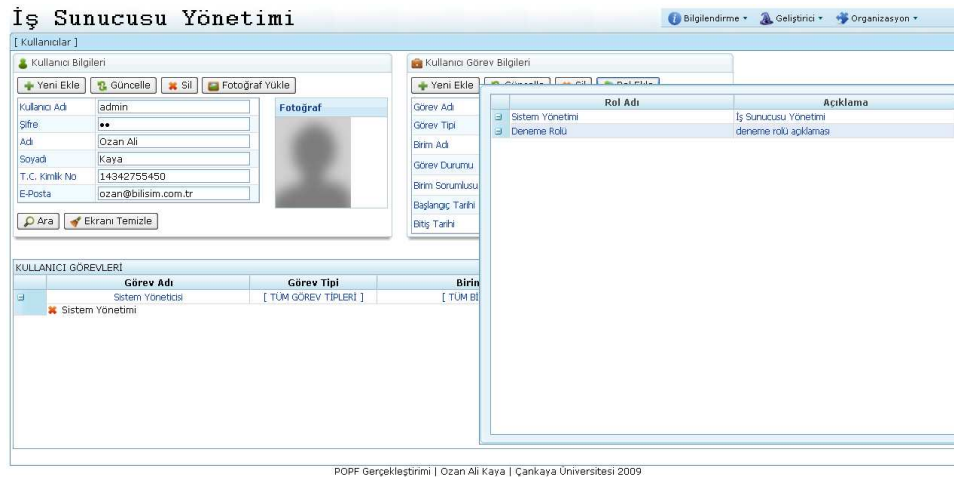


Figure 5.17: Organization and Authorization System, Add Role to User

CHAPTER 6

CONCLUSION

In this thesis, the role of productivity in creation of software products was examined. It has been addressed both what the impact of the productivity on the cost of software products is and how the success of the software companies in dealing with productivity increases.

Productivity, is not only the result of the correct application of the technique but also a requirement to keep the software up to date and cheap.

In the case of software development methodology, Productivity Oriented Programming is similar to Agile Programming. Product-oriented approach, however, is agile and aims sustainable success. With this aspect, POP is thought to be the correct choice for software companies.

POPF is intended to be used by non-qualified programmers. A complex project's sample module is designed and implemented with POPF by a couple of non-qualified programmers within a week. Thus, simple structure of POPF simplified the adaptation of newly hired developers to the software team. The time spent is nearly the quarter of the time for regular methods.

6.1 Future Works

The continuation of this work is considered as the implementation of the requirements which were specified but not implemented. In addition, the rule based operating segments which are meant to ease the coding will be designed to deal with the domain knowledge. With this way, it is aimed to increase the manageability and the quality of the source code of applications.

Defining rules is the common way to describe domain knowledge in algorithms. If rule struc-

ture can be simplified as to be viewed and managed in user interfaces then it will be possible for the domain experts act like developers. By the time, rules change with requirements and the application normalizes. Performance of the application depends on your infrastructure team's success. So, software companies stop educating their developers as domain experts.

Rule based coding does not need to be deployed. All system modules of POPF should be written in pure Java. But if domain codes written in rules, that would be great support for non-stop application execution aim of POPF.

Hardware requirements of software companies are really great. If POPF had a web-based GUI to be used as IDE, developers would develop their rule-based services over the web with cheap computers. Location problem of developers would also be solved by this way.

REFERENCES

- [1] http://en.wikipedia.org/wiki/Service-oriented_architecture
- [2] **Arifoğlu A.** (2004), *e-Dönüşüm: Yol Haritası, Türkiye, Dünya*, Sas Bilişim, Ankara
- [3] **Brooks F.** (1995), *The Mythical Man-Month: Essays on Software Engineering*, Addison-Wesley
- [4] <http://jpa2web.sourceforge.net/>
- [5] <http://www.ibm.com/developerworks/web/library/wa-aj-jpa2web/>
- [6] <http://www.gestion400.com/web/guest/home>
- [7] <http://sourceforge.net/projects/jseahorse/>
- [8] <http://seyhanbasmaci.blogspot.com/>
- [9] <http://stripesframework.org/display/stripes/Home>
- [10] <http://www.springsource.org/>
- [11] <http://seamframework.org/>
- [12] <http://www.ics.uci.edu/~wscacchi/Presentations/Process/Software-Productivity.ppt>
- [13] <http://java.sun.com/developer/technicalArticles/J2EE/jpa/>
- [14] <https://www.hibernate.org/>
- [15] <http://www.oracle.com/technology/products/ias/toplink/index.html>
- [16] <http://java.sun.com/jdo/index.jsp>
- [17] <http://www.mysql.com>
- [18] <https://jaxb.dev.java.net>
- [19] <http://ws.apache.org/jaxme>
- [20] <http://www.csg.is.titech.ac.jp/~chiba/javassist/>
- [21] <http://scannotation.sourceforge.net>
- [22] <http://jasperforge.org/>
- [23] http://java.sun.com/j2ee/tutorial/1_3-fcs/doc/EJBQL.html
- [24] <http://docs.jboss.org/hibernate/stable/core/reference/en/html/queryhql.html>