T.C. DOĞUŞ UNIVERSITY
INSTITUTE OF SCIENCE & TECHNOLOGY
Computer Information Sciences

Service Oriented Architecture

Master Thesis

Murat Kaya
200691001

Professor Dr. Selim Akyokus

January 2009
Istanbul

ACKNOWLEDGEMENTS

I extend my sincere gratitude and appreciation to many people who made this master thesis possible. Special thanks are due to my supervisor Professor Selim Akyokus, Professor Izzet Cem Goknar and Professor Coskun Sonmez. I appreciate the confidence that my supervisor showed in my abilities and (especially) the patience he exhibited in waiting for me to get things done.

I am highly indebted to the colleagues from Turkcell Technology Company just for accepting me to carry on my research at the hard-working hours. They gave me different visions for handling problems ranging from minor to major.

Many thanks go to frist of all my twin brother Cem Kaya and one of my dearest friend Yasin Ozan Kilic from the Computer Engineering Department of Isik University. They shared my problems and gave me strength with their valuable advices.

Special thanks go to my family since they showed me the value of hard work and always encouraged me to set my goals high. Especially, my mother who believes me more than anyone else never drawbacks her spiritual support ignoring her own problems. If this thesis satisfies its goals, the main reason is her eternal trust and encouragements.

Last but not the least, I would like to thank to the most Gracious and Almighty creator of everything. He blesses me with everything I own.

Istanbul, January 2009 Murat KAYA

ABSTRACT

I present a survey that describes service oriented architecture. I made an sample application on Oracle BPEL by using Oracle SOA Suite for demonstration issues using it. I made researches about properties of service oriented architecture, advantages and disadvantages of it. Besides, the past studies that were made by using SOA has been analyzed. In this thesis principles of service oriented architecture are explained.

Service oriented architecture can be used in different areas by the help of its properties and advantages. It can be adopted to different platforms easily. The use of SOA in applications decreases the development time but making services at the beginning is not good in terms effort. The main advantage of service oriented architecture is reusability, a service can be used in different applications and there is no need to make any changes on the service. To have the description of any service is adequate to use it. A service can be used by sending the necessary parameters to the service.

In application part, an application has been developed step by step by using the properties of service oriented architecture.

ÖZET

Servis Tabanlı Mimari(Service Oriented Architecture) hakkında bir araştırma sunuyorum. Oracle SOA Suite'i kullanarak anlattıklarımı desteklemek için örnek bir uygulama geliştirdim. Servis tabanlı mimarinin özellikleri, avantajları ve dezavantajları hakkında araştırma yaptım. Bunların yanı sıra daha önce bu teknoloji kullanılarak yapılan araştırmaları inceledim. Bu tezde servis tabanlı mimarinin temel özellikleri ve prensipleri anlatılmıştır.

Servis tabanlı mimari özellikleri ve avantajları sayesinde çok değişik alanlarda kullanılmaktadır. Değişik platformlara çok kolaylıkla adapte edilebilmektedir. Servis tabanlı mimariyi uygulamalarda kullanmak geliştirme aşamasında zaman açısından çok büyük avantajlar sağlar, ancak sürecin başlangıcında servisleri oluşturmak zaman açısından çok avantajlı değildir. Servis tabanlı mimarinin en belirgin özelliği tekrar kullanılabilirliğidir, bir servis hiçbir değişiklik yapılmadan değişik uygulamalarda rahatlıkla kullanılabilir. Bir servisin açıklamasına sahip olmak onu kullanmak için yeterlidir. Bir servisi kullanmak için gerekli değişkenleri göndermek yeterlidir.

Uygulama kısmında servis tabanlı mimarinin özellikleri kullanılarak adım adım bir uygulama geliştirilmiştir.

TABLE OF CONTENTS

Sayfa

ACKNOWLEDGEMENTS ii

ABSTRACT iii

ÖZET iv

TABLE OF CONTENTS v

LIST OF FIGURES viii

1. INTRODUCTION 1

1.1 Encapsulation Logic 1
1.2 Service Relation 2
1.3 Service Communication 3
1.4 Service Design 4
1.5 SOA 5

2. PROPERTIES OF SOA 6

2.1 Core of the service-oriented computing platform 6
2.2 Quality of Service 6
2.3 Autonomous 7
2.4 Based on open standards 7
2.5 Promotes vendor diversity 8
2.6 Supports discovery 8
2.7 Supports interoperability 9
2.8 Supports federation 9
2.9 Supports composability 10
2.10 Supports reusability 10
2.11 Underlines extensibility 11
2.12 Supports a service-oriented business modeling 11
2.13 Implements layers of abstraction 12
2.14 Supports loose coupling 12
2.15 Supports organizational agility 13
2.16 SOA 14

3. BENEFITS OF SOA 16

3.1 Improved integration 16
3.2 Inherent reuse 16
3.3 Regular architectures and solutions 16
3.4 Leveraging the legacy investment 16
3.5 Establishing standardized XML 17
3.6 Focused investment on communications infrastructure 17
3.7 “Best-of-breed” alternatives 17
3.8 Organizational agility 17

4. COMMON DISADVANTAGES 19

4.1 Building like traditional distributed architectures 19
4.2 Not standardizing SOA 19
4.3 Not to have a transition plan 20

4.4 Not starting with an XML foundation architecture ... 21
4.5 Not figure out SOA performance requirements ... 21
4.6 Not understanding Web services security ... 22
4.7 Not keeping in touch developing technology ... 22

5. USAGE OF SOA ... 23

5.1 In e-education systems ... 23
5.2 In military communication networks ... 25
5.3 SOA with grid computing ... 27
5.4 SOA Solution Providers ... 30
5.4.1 Microsoft BizTalk Server ... 30
5.4.2 IBM WebSphere ... 31
5.4.3 TIBCO ... 31
5.4.4 SUN J2EE ... 32
5.4.5 ORACLE SOA SUITE ... 33

6. ORACLE SOA ... 34

6.1 Description of Oracle SOA ... 34
6.1.1 Integrated Service Environment ... 35
6.1.2 Oracle BPEL Process Manager ... 35
6.1.3 Oracle Enterprise Service Bus (ESB) ... 36
6.1.4 Oracle Business Rules ... 36
6.1.5 OracleAS Integration Business Activity Monitoring ... 37
6.1.6 Oracle Web Services Manager ... 37
6.1.7 OracleAS UDDI Registry ... 37
6.1.8 Oracle Application Server ... 37

7. SOA Design & Implementation of Online Campus ... 38

7.1 Web Services ... 40
7.1.1 GetLoginAccount ... 40
7.1.2 GetStudentGrades ... 40
7.1.3 GetStudentSelectedCourses ... 40
7.1.4 GetStudentSelectedCoursesNumber ... 40
7.1.5 GetStudentSelectedCoursesNumberOfConflicts ... 40
7.1.6 GetStudentSelectedCoursesTotalCredit ... 40
7.1.7 GetStudentsWaitingForApproval ... 40
7.1.8 GradeTheStudent ... 40
7.1.9 IsStudentProgramApproved ... 40
7.1.10 LendBookToStudent ... 41
7.1.11 ReturnBook ... 41
7.1.12 UpdateBook ... 41
7.1.13 UpdateCourse ... 41
7.1.14 UpdateDepartment ... 41
7.1.15 UpdateInstructor ... 41
7.1.16 UpdateLogin ... 41
7.1.17 UpdateStudent ... 41
7.1.18 GetStudentGPA ... 41
7.1.19 AddStudentFinancialHold ... 41
7.1.20 RemoveFinancialHold ... 41
7.1.21 GetStudentSemesterGPA ... 41
7.1.22 IsStudenEligibleForRegistration Service ... 42
7.1.23 GetStudentSubmittedCoursesInformation Service ... 43

7.1.24 GetStudentInformation Service 44
7.1.25 GetStudentAddressInformation Service 45
7.1.26 AddStudentAddressInformation Service 45
7.1.27 GetCourseInformation Service 46
7.1.28 GetStudentSelected Courses Service 47
7.1.29 GetInstructor Service 47
7.1.30 DisapproveStudentProgram Service 48
7.1.31 GetCoursesOffered Service 49
7.1.32 AddBook Service 49
7.1.33 AddCourse Service 49
7.1.34 AddDepartment Service 49
7.1.35 AddFaculty Service 50
7.1.36 AddStudent Service 50
7.1.37 AddStudentCourse 50
7.1.38 AddStudentPenalty Service 50
7.1.39 ApproveStudentProgram service 51
7.1.40 CreateLogin Service 51
7.1.41 DeleteBook Service 51
7.1.42 DoesStudentHaveDept Service 51
7.1.43 DoesStudentHavePenalty Service 51
7.1.44 DoesStudentHaveUnreturnedBooks Service 51
7.1.45 DropSelectedCourse Service 51
7.1.46 GetCourseStudents Service 52
7.1.47 GetInstructorCourses 52
7.1.48 GetLogin Service 52
7.1.49 GetPenalties Service 52
7.1.50 GetStudentFinalCourses Service 52
7.1.51 GetStudentFinalGrades Service 53
7.1.52 SubmitCourses Service 53

8. CONCLUSION 54

REFERENCES 57

Appendix I. IsStudentEligibleForRegistration BPEL 60

Appendix II. GetCoursesOfferedResponse XSL Ttransformation 64

Appendix III. An example Oralce SOA generated WSDL 66

CURRICULUM VITAE 67

LIST OF FIGURES

Figure 1-1 Services can encapsulate varying amounts of logic.. 2
(Eric Newcomer and Greg Lomow, 2004)... 2
Figure 1-2 Service description for B... 3
(Munindar P. Singh and Michael N. Huhns, 2005) 3
Figure 1-3 A message existing as an independent unit of communication............................ 4
(Eric Newcomer and Lomow, 2004) 4
Figure 1-4 Service-orientation principles address design issues.. 4
(Munindar P. Singh and Michael N. Hhns, 2005) 4
Figure 2-1 Contemporary SOA is based on open standards 8
(Munindar P. Singh and Michael N. Huhns, 2005) 8
Figure 2-2 Disparate technology platforms.. 8
(Munindar P. Singh and Michael N. Huhns, 2005) 8
Figure 2-3 Registries enable a mechanism for the discovery of services. 9
(Munindar P. Singh and Michael N. Huhns, 2005) 9
Figure 2-4 Services enable unforeseen integration opportunities... 9
(Munindar P. Singh and Michael N. Huhns, 2005) 9
Figure 2-5 Services enable standardized federation. 10
(Munindar P. Singh and Michael N. Huhns, 2005) 10
Figure 2-6 Different solutions can be composed. (Eric Newcomer and Greg Lomow, 2004) 10
Figure 2-7 Inherent reuse accommodates unforeseen reuse opportunities. 11
(Eric Newcomer and Greg Lomow, 2004)... 11
Figure 2-8 Extensible services can expand functionality.. 11
(Eric Newcomer and Greg Lomow, 2004)... 11
Figure 2-9 A collection of services encapsulating business process logic............................ 12
(Eric Newcomer and Greg Lomow, 2004)... 12
Figure 2-10 Application logic can be abstracted through a dedicated layer. 12
(Eric Newcomer and Greg Lomow, 2004)... 12
Figure 2-11 Service layers that abstract business and application logic.............................. 13
(Eric Newcomer and Greg Lomow, 2004)... 13
Figure 2-12 A loosely coupled relationship allows efficient respond to changes. 14
(Eric Newcomer and Greg Lomow, 2004)... 14
Figure 4.1 High-level view of e-education system architecture.. 24
Figure 4.2 Results of different compression techniques. 26
Figure 4.3 OGSA-DAI data interface 28

Figure 4.4 OGSA-DQP architecture 29
Figure 4.5 Response times for analysis queries with different levels of parallelism............. 30
(Steven Lynden et al., 2008).. 30
Figure 6.1 Oracle SOA Suite Architecture(Oracle Soa, 15 October 2008) 35
Figure 7.1. Online Campus Database 39
Figure 7.2.a IsStudentEligibleForRegistration 42
Figure 7.2.b IsStudentEligibleForRegistration 43
Figure 7.2.c IsStudentEligibleForRegistration 43
Figure 7.3. GetStudentSubmittedCourseInformation BPEL Process design 44
Figure 7.4. GetStudentInformation BPEL process design 44
Figure 7.5. GetStudentAddressInformation BPEL Process design 45
Figure 7.6. AddStudentAddressInformation BPEL Process design 46
Figure 7.7. GetCourseInformation BPEL Process Design 46
Figure 7.8. GetStudentSelectedCourses BPEL Process design 47
Figure 7.9. GetInstructor BPEL Process design 48
Figure 7.10. DisapproveStudentProgram BPEL design... 48
Figure 7.11. GetCoursesOffered XSLT Design 49
Figure 7.12. AddStudent input transformation assignment.. 50
Figure 7.13. AddStudentCourse XSLT Design 51
Figure 7.14. GetCourseStudent XSLT Design. 52
Figure 7.15. Design of assignment in Oracle SOA Suite... 53
Figure 7.16. Webservice call setup window for GetPenalties 53

1. INTRODUCTION

The term “service-oriented” offers a distinct aproach for separating concerns. What this means is that logic required to solve a large problem can be better understood, constructed, carried out and managed if it is divided into a collection of smaller, related pieces. Each of these pieces addresses a concern or a specific part of the big problem at the start point. We can think this logic as a divide and conquer method for solving bigger problems by dividing it into smaller parts.

“Service-oriented architecture” is a term that represents a model in which automation logic is separated into smaller, distinct units of logic. If we aggregare these smaller units a larger piece of business automation logic is comprised. Individually, these units can be distributed.

Although in a distributed business community, we could inhibit the potential of individual businesses, if we impose oppressive subjection. Yet we want to allow outlets to interact and heaver each other’s services, we want to avoid a model in which outlets form tight connections that result in constrictive inter-dependencies.

Service-oriented architecture (SOA) supports individual units of logic to exist autonomously but not isolated from each other. While still maintaining a adequate amount of commonality and standardization, units of logic are still required to conform to a set of principles which allow them to evolve independently. These units of logic are known as services within SOA.

1.1 Encapsulation Logic

To support independency, services encapsulate logic within a distinct context. This context can be specific to a business task, a business entity, or some other logical grouping.

As shown in Figure 1.1, when building an automation solution consisting of services, each service can encapsulate a task performed by an individual step or a sub-process comprised of a set of steps. A service can even encapsulate the entire process logic. In the latter two cases, the larger scope represented by the services may contains the logic encapsulated by other services.

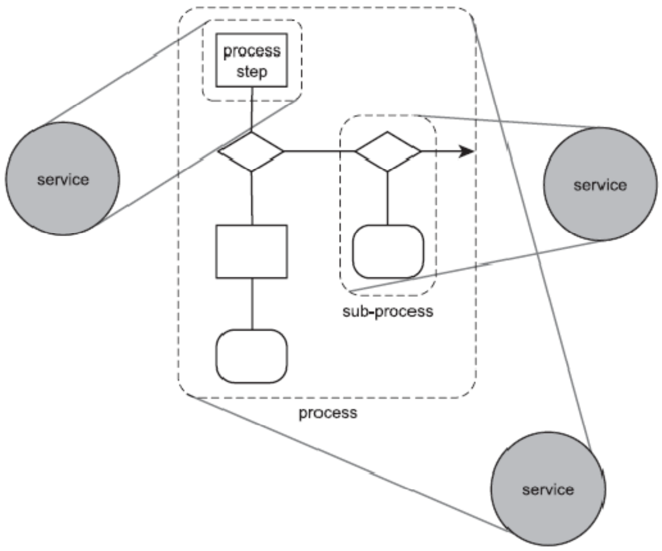


Figure 1-1 Services can encapsulate varying amounts of logic.
(Eric Newcomer and Greg Lomow, 2004)

For services to use the logic they encapsulate they can participate in the execution of business activities. In this manner, they must establish distinct relationships with the services that want to use them.

1.2 Service Relation

In service oriented architecture, services can be used by other services or other programs. Two services must be aware of each other if they want to communicate. This awareness is achieved through the use of service descriptions. This relationship is based on an understanding of their service descriptions.

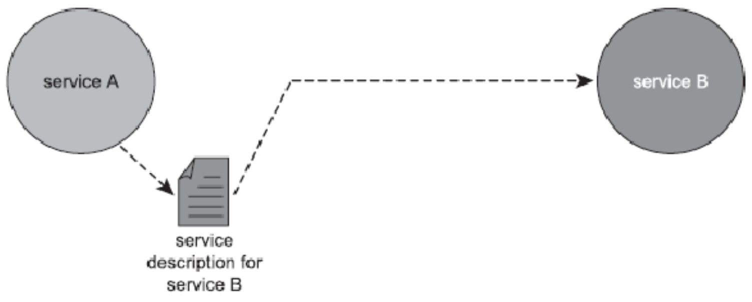


Figure 1-2 Service description for B.
(Munindar P. Singh and Michael N. Huhns, 2005)

A service description includes basicly the name of the service and the data expected and returned by the service. The manner in which services use service descriptions results in a relationship classified as loosely coupled. For example, Figure 1.2 illustrates that service A is aware of service B because service A is in possession of service B’s service description.

For services to interact and accomplish something meaningful, they must exchange information. A communications method is required to supporttheir loosely coupled relationship. One such method is messaging.

1.3 Service Communication

After a service sends a message on its way, it doesn’t have any control on the message anymore and it doesn’t know what happens to the message. So messages must be independent units of communication. This means that messages should be autonomous. To that effect, messages can be outfitted with enough intelligence to self-govern their parts of the processing logic (Figure 1.3).

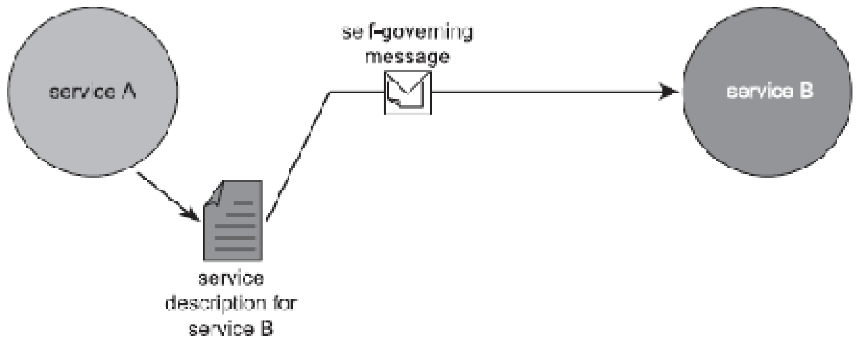


Figure 1-3 A message existing as an independent unit of communication.
(Eric Newcomer and Lomow, 2004)

Services that provide service descriptions and communicate via messages form a basic architecture. What distinguishes ours is how its three core components (services, descriptions, and messages) are designed. This is where service-orientation comes in.

1.4 Service Design

It is like an object-orientation. Service-orientation has become a diverse design approach that establishes commonly accepted principles which govern the positioning and design of our architectural components (Figure 1.4).

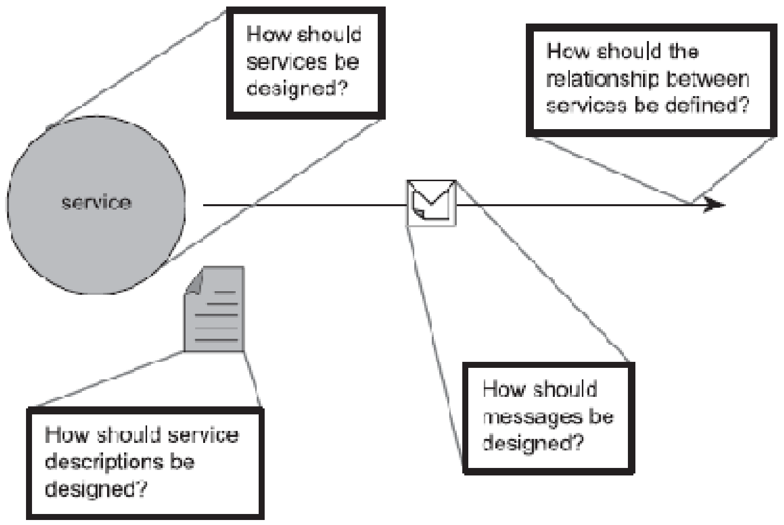

How should services be designed?
How should the relationship between services be defined?
service
How should service descriptions be designed?
How should messages be designed?

Figure 1-4 Service-orientation principles address design issues.
(Munindar P. Singh and Michael N. Hhns, 2005)

The application of service-orientation principles to processing logic results in standardized service-oriented processing logic. When a solution is covered by units of service oriented processing logic, it becomes a service-oriented solution.

•**Loose coupling:** Services maintain a relationship that minimizes dependencies and only requires that they retain an awareness of each other.

•**Service contract:** Services adhere to a communications agreement, as defined collectively by one or more service descriptions and related documents.

• **Autonomy:** Services have control over the logic they encapsulate.

• **Abstraction:** Beyond what is described in the service contract, services hide logic from the outside world.

• **Reusability:** Logic is divided into services with the intention of promoting reuse.

•**Composability:** Collections of services can be coordinated and assembled to form composite services.

• **Statelessness:** Services minimize retaining information specific to an activity.

• **Discoverability:** Services are designed to be outwardly descriptive so that they can be found and assessed via available discovery mechanisms.(Axel Buecker et al, 2007)

1.5 SOA

The past few sections have described the individual ingredients for what we call primitive SOA. It is classified alike because it represents a substratum technology architecture that is supported by current major vendor platforms.

All forms of SOA we explore from here on are based on and extend this primitive model. Some of the extensions we discuss are attainable today through the application of advanced design techniques, while others rely on the availability of pre-defined Web services specifications and corresponding vendor support.

2. PROPERTIES OF SOA

The look of Service Oriented Architecture is shaped by the ongoing industry trends and developmets. Its founding principles remain, but many have been expanded primarily because the opportunity to do so has been readily acted upon.

Major software vendors are continually proposing new Web services specifications and creating increasingly powerful XML and Web services support into current technology platforms. The result is an extended variation of service-oriented architecture. Which is named contemporary SOA.

Contemporary SOA establishes upon the primitive SOA model by reusing industry and technology advancements to further its original ideals. Though the required implementation technology can vary, contemporary SOAs have evolved to a point where they can be associated with a set of common characteristics.

2.1 Core of the service-oriented computing platform

Many argue that the manner in which SOA is used to qualify products, designs, and technologies elevates this term beyond one that simply relates to architecture.

Past terms used to identify distinct application computing platforms were often suffixed with the word “architecture” when the architecture was actually being referenced. The terms “client-server” or “n-tier,” for example, can be used to classify a tool, an administration infrastructure, or an application architecture. With SOA, however, the actual acronym has become a multi-purpose buzzword used frequently when discussing an application computing platform consisting of Web services technology and service-orientation principles

“Contemporary SOA represents an architecture that promotes service-orientation through the use of Web services.” (Axel Buecker et al., 2007)

2.2 Quality of Service

• The ability for tasks to be carried out in a secure manner, protecting the contents of a message, as well as access to individual services.

• Allowing tasks to be carried out reliably so that message delivery or notification of failed delivery can be guaranteed.

• Performance requirements to ensure that the overhead imposed by SOAP message and XML content processing does not inhibit the execution of a task.

• Transactional capabilities to protect the integrity of specific business tasks with a guarantee that should the task fail, exception logic is executed.

Contemporary SOA fills the QoS gaps of the primitive SOA model. For lack of a better term, we'll refer to an SOA that fulfills specific quality of service requirements as "QoS-capable." (Munindar P. Singh and Michael N. Huhns, 2005)

2.3 Autonomous

The service-orientation principle of autonomy requires that individual services must be independent and self-contained. This is further realized through message-level autonomy where messages passed between services are sufficiently intelligence-heavy that they can control the manner in which they are processed by recipient services. SOA builds upon and expands this principle by promoting the concept of autonomy throughout solution environments and the enterprise.

2.4 Based on open standards

The most significant characteristic of Web services is the truth that data exchange is governed by open standards. After a message is sent from one Web service to another it travels via a set of protocols that is globally standardized and accepted.

Further, the message itself is standardized, both in format and in how it represents its payload. The use of SOAP, WSDL, XML, and XML Schema allow for messages to be fully self-contained and support the underlying agreement that to communicate. As we mentioned before services only need to know each other's service descriptions. The use of an open, standardized messaging model eliminates the need for underlying service logic to share type systems and supports the loosely coupled paradigm.

Contemporary SOAs fully leverage and reinforce this open, vendor-neutral communications framework (Figure 2-1).

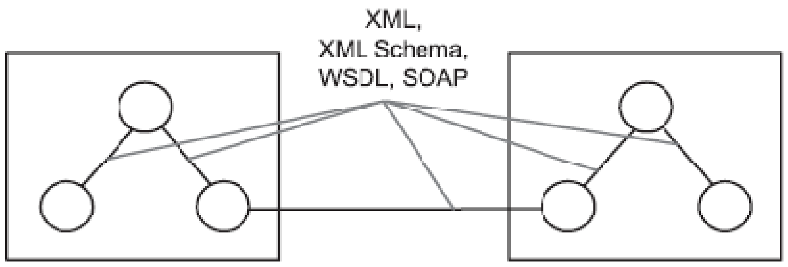


Figure 2-1 Contemporary SOA is based on open standards
(Munindar P. Singh and Michael N. Huhns, 2005)

2.5 Promotes vendor diversity

The open communications method that I have explained in above section not only has significant suggestions for bridging much of the heterogeneity within corporations, but it also allows organizations to choose best-of-breed environments for specific applications.

For example, regardless of how individual a development environment is, as long as it supports the creation of standard Web services, it can be used to create a non-proprietary service interface layer, opening up interoperability opportunities with other, service capable applications (Figure 2-2).

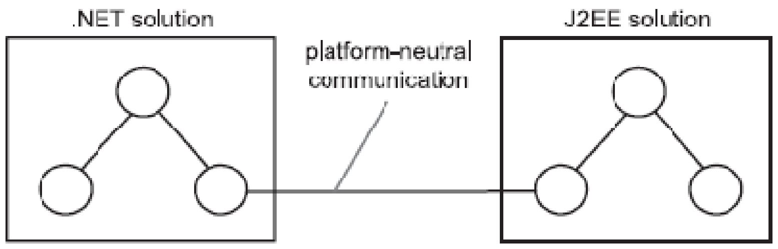


Figure 2-2 Disparate technology platforms.
(Munindar P. Singh and Michael N. Huhns, 2005)

2.6 Supports discovery

Web services were frequently used to facilitate point-to point solutions, when utilized within traditional distributed architectures. Consequently, discovery was not a common interest.

SOA supports and encourages the advertisement and discovery of services throughout the enterprise and beyond. A serious SOA will likely rely on some form of service registry or directory to manage service descriptions (Figure 2-3).

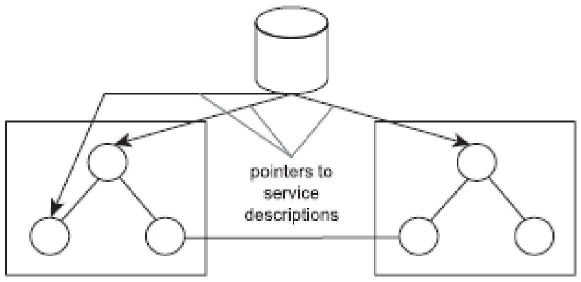


Figure 2-3 Registries enable a mechanism for the discovery of services.
(Munindar P. Singh and Michael N. Huhns, 2005)

2.7 Supports interoperability

When building an SOA application from the scratch, services with intrinsic interoperability become potential integration endpoints (Figure 2-4). When properly standardized, this leads to service-oriented integration architectures where in solutions themselves obtain a level of interoperability. Supporting this characteristic can significantly decrease the cost and effort of fulfilling future cross-application integration requirements.

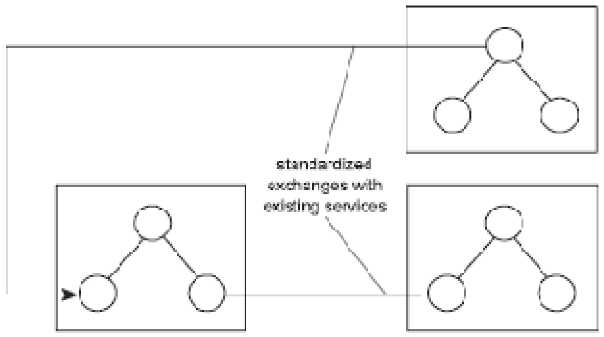


Figure 2-4 Services enable unforeseen integration opportunities.
(Munindar P. Singh and Michael N. Huhns, 2005)

2.8 Supports federation

Establishing SOA within an enterprise does not necessarily require that you replace what you already have.

Obviously, the incorporation of SOA with previous platforms can lead to a variety of hybrid solutions. However, the key aspect is that the communication channels achieved by this form of service-oriented integration are all uniform and standardized (Figure 2-5).

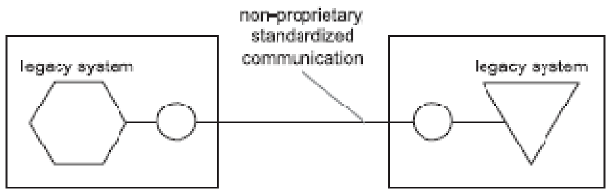


Figure 2-5 Services enable standardized federation.
(Munindar P. Singh and Michael N. Huhns, 2005)

2.9 Supports composability

Composability is a deep-rooted characteristic of service oriented architecture that can be realized on different levels. As we mentiones in previous sections, services acts as an independent units of logic. A business process can therefore be divided into a series of services, each responsible for executing a little part of the whole process(Figure 2-6).

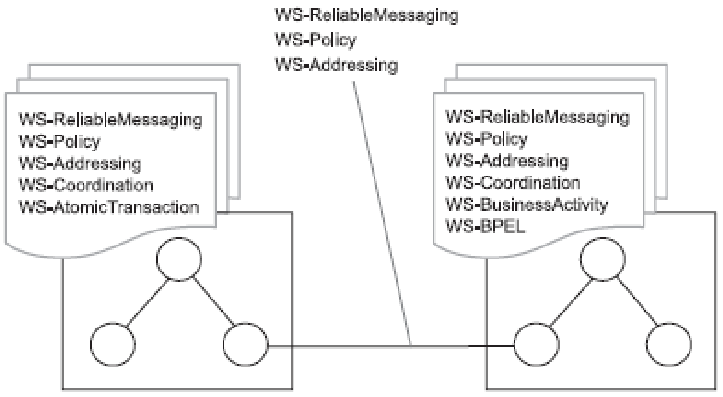


Figure 2-6 Different solutions can be composed.
(Eric Newcomer and Greg Lomow, 2004)

2.10 Supports reusability

SOA establishes an environment that promotes reuse on many levels. The emphasis placed by SOA on the creation of services which are standalone to both the automation solutions and the business processes that utilize them leads to an environment in which reuse is naturally realized as a side benefit to delivering services for a given project. Thus, inherent reuse can be supported when building service-oriented solutions (Figure 2-7).

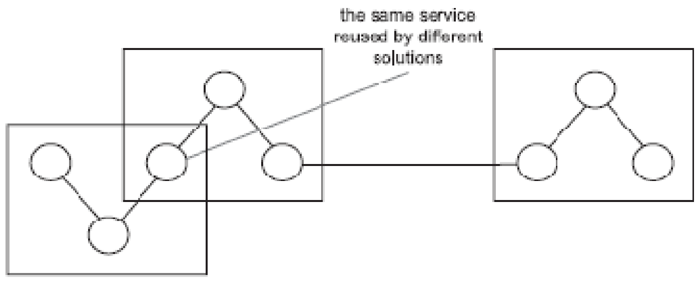


Figure 2-7 Inherent reuse accommodates unforeseen reuse opportunities.
(Eric Newcomer and Greg Lomow, 2004)

2.11 Underlines extensibility

When service logic is properly partitioned into small pieces, the scope of functionality offered by a service can sometimes be extended without breaking the established interface (Figure 2-8).

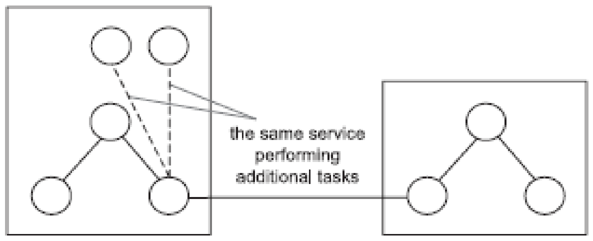


Figure 2-8 Extensible services can expand functionality.
(Eric Newcomer and Greg Lomow, 2004)

At this point we can make the definition of contemporary SOA as follows:

SOA "represents an open, extensible, federated, composable architecture that promotes service-orientation and is comprised of autonomous, QoS-capable, vendor diverse, interoperable, discoverable, and potentially reusable services, implemented as Web services." (Eric Newcomer and Greg Lomow, 2004)

2.12 Supports a service-oriented business modeling

As we mentioned before in our description of a primitive SOA, we briefly explored how business processes can be represented and expressed through services. Partitioning business logic into services that can be composed has significant implications as to how business processes can be modeled (Figure 2-9). Analysts can leverage these features by incorporating an extent of service-orientation into business processes for implementation through SOAs.

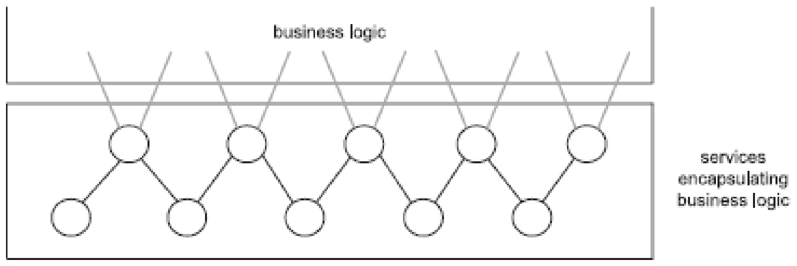


Figure 2-9 A collection of services encapsulating business process logic.
(Eric Newcomer and Greg Lomow, 2004)

2.13 Implements layers of abstraction

One of the characteristics of service-oriented design principles is that of abstraction. Typical SOAs can introduce layers of abstraction by positioning services as the only access points to a variety of resources and processing logic.

When applied through proper design, abstraction can be targeted at business and application logic(Figure 2-10). The only remaining concern is the functionality offered via the service interfaces.

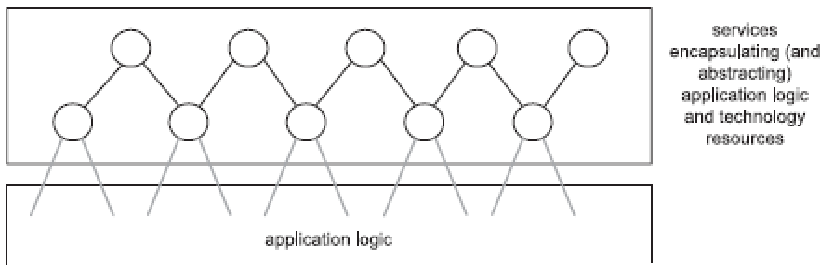

services encapsulating (and abstracting) application logic and technology resources
application logic

Figure 2-10 Application logic can be abstracted through a dedicated layer.
(Eric Newcomer and Greg Lomow, 2004)

2.14 Supports loose coupling

By implementing standardized service abstraction layers, a loosely coupled relationship also can be achieved between the business and application technology domains of an enterprise

(Figure 2-11). Each end only requires an awareness of the other, therefore allowing each domain to evolve more independently. The result is an environment that can better accommodate business and technology-related change a quality known as organizational agility.

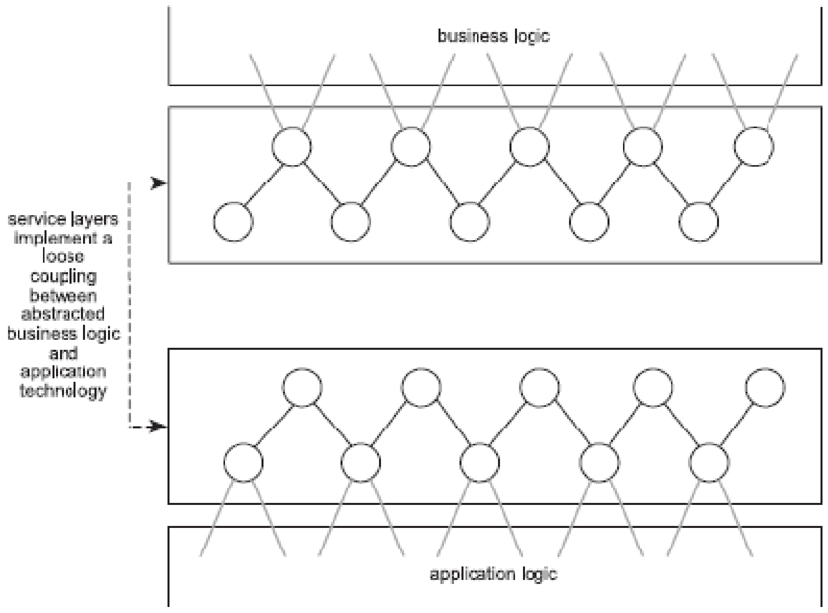

business logic
service layers implement a loose coupling between abstracted business logic and application technology
application logic

Figure 2-11 Service layers that abstract business and application logic.
(Eric Newcomer and Greg Lomow, 2004)

2.15 Supports organizational agility

Any change in an service's business logic can impact the application technology that automates it. Change in an service's application technology infrastructure can impact the business logic automated by this technology. The more dependencies that exist between these two parts of an enterprise, the greater the extent to which change imposes disruption and expense.

By influencing service business representation, service abstraction, and the loose coupling between business and application logic supplied through the use of service layers, SOA offers the potential to increase organizational agility (Figure 2-12).

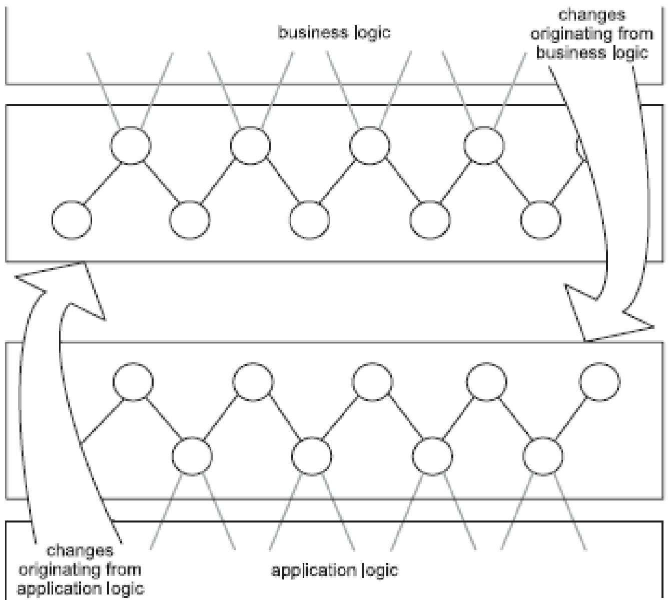

business logic
changes originating from business logic
changes originating from application logic
application logic

Figure 2-12 A loosely coupled relationship allows efficient respond to changes. (Eric Newcomer and Greg Lomow, 2004)

While the characteristics that we descirbed up to this point are fundamental to contemporary SOA, this point is obviously more of a subjective statement of where SOA is at the moment. Even though SOA is being positioned as the next standard application computing platform, this transition is not yet complete. Despite the fact that Web services are being used to implement a great deal of application functionality, the support for a number of features necessary for enterprise-level computing is not yet fully available.

2.16 SOA

Now that we've finished covering characteristics, we can finalize our formal definition.

"Contemporary SOA represents an open, agile, extensible, federated, composable architecture comprised of autonomous, QoS-capable, vendor diverse, interoperable, discoverable, and potentially reusable services, implemented as Web services.

SOA can establish an abstraction of business logic and technology that may introduce changes to business process modeling and technical architecture, resulting in a loose *coupling between these models.*

SOA is an evolution of past platforms, preserving successful characteristics of traditional architectures, and bringing with it distinct principles that foster service-orientation in support of a service-oriented enterprise.

SOA is ideally standardized throughout an enterprise, but achieving this state requires a planned transition and the support of a still evolving technology set." (Eric Newcomer and Greg Lomow, 2004)

Though accurate, this definition of contemporary SOA is quite detailed. For practical purposes, let's provide a supplementary definition that can be applied to both primitive and contemporary SOA.

"SOA is a form of technology architecture that adheres to the principles of service-orientation. When realized through the Web services technology platform, SOA establishes the potential to support and promote these principles throughout the business process and automation domains of an enterprise." (Munindar P. Singh and Michael N. Huhns, 2005)

3. BENEFITS OF SOA

SOA will benefit organizations in different ways, depending on their respective goals and the manner in which SOA and its supporting type of products and technologies is applied. This list of common benefits is generalized and certainly not complete. It is merely an indication of the potential this architectural platform has to offer.

3.1 Improved integration

SOA can result in the creation of solutions that consist of inherently interoperable services. Utilizing solutions based on interoperable services is part of service-oriented integration and results in a service-oriented integration architecture.

Because of the vendor-neutral communications framework the potential is there for enterprises to implement highly standardized service descriptions and message structures. The net result is intrinsic interoperability, which turns a cross-application integration project into less of a custom development effort, and more of a modeling exercise.

3.2 Inherent reuse

Service-orientation supports the design of services that are inherently reusable. Building services to be inherently reusable results in a moderately increased development effort and requires the use of design standards. Subsequently leveraging reuse within services lowers the cost and effort of building service-oriented solutions.

3.3 Regular architectures and solutions

The concept of composition is another fundamental part of SOA. This aspect of service-oriented architecture can lead to highly optimized automation environments. Benefits of streamlined solutions and architectures include the potential for reduced processing overhead and reduced skill-set requirements.

3.4 Leveraging the legacy investment

Web services technology set has spawned a large adapter market, enabling many legacy environments to participate in service-oriented integration architectures. The ability to use what you already have with service-oriented solutions that you are building now and in the future is extremely attractive.

3.5 Establishing standardized XML

SOA is text based and built upon and driven by XML. As a result, an adoption of SOA leads to the opportunity to fully leverage the XML data representation platform. A standardized data representation format can decrease the underlying complexity of all affected application environments.

Past efforts to standardize XML technologies have resulted in limited success. These approaches severely inhibited the potential benefits XML could introduce to an organization. With contemporary SOA, establishing an XML data representation architecture becomes a necessity, providing organizations the opportunity to achieve a broad level of standardization.(M. Gerz et al., 2006)

3.6 Focused investment on communications infrastructure

This allows organizations to evolve enterprise-wide infrastructure by investing in a single technology set responsible for communication. The cost of scaling communications infrastructure is reduced, as only one communications technology is required to support the federated part of the enterprise.

3.7 "Best-of-breed" alternatives

Most of the critiques against IT departments are related to the restrictions imposed by a given technology platform on its ability to fulfill the automation requirements of an organization's business areas. SOA won't solve these problems entirely, but it is expected to increase empowerment of both business and IT communities. One of the key features of service-oriented enterprise environments is the support of "best-of-breed" technology. It frees IT departments from being chained to a single proprietary development and/or middleware platform. For any given piece of automation that can expose an adequate service interface, you now have a choice as to how you want to build the service that implements it.

3.8 Organizational agility

Agility means adopting the ability to adopt itself to the environmental changes. Agility is a quality inherent in just about any aspect of the enterprise. Much of service-orientation is based on the assumption that what you build today will evolve over time. One of the primary benefits of a well-designed SOA is to protect organizations from the impact of this evolution.

Change can be disruptive, expensive, and potentially damaging to inflexible IT environments. A standardized technical environment comprised of loosely coupled, composable, interoperable and potentially reusable services establishes a more adaptive automation environment that empowers IT departments to more easily adjust to change. By abstracting business logic and technology into specialized service layers, SOA can establish a loosely coupled relationship between these two enterprise domains. This allows each domain to evolve independently and adapt to changes imposed by the other, as required.

4. COMMON DISADVANTAGES

Considering the extent to which organizations need to shift technology and mindset to fully adopt SOA, it is actually probable that some will inadvertently build bad service-oriented architectures.

4.1 Building like traditional distributed architectures

The most common difficulty for organizations is building traditional distributed architectures under the pretense that they are building SOA. As a result of this an organization can go quite far in terms of integrating the Web services technology set before realizing that they've been heading down the wrong path.

Examples of some of the problems this can introduce include:

- Reproduction of RPC-style service descriptions .
- Restricting the adoption of features provided by WS-* specifications.
- Wrong partitioning of functional boundaries within services.
- Creating non-composable (or semi-composable) services.
- Further entrenchment of synchronous communication patterns.
- Creating hybrid or non-standardized services.

Understanding the fundamental differences between SOA and previous architectures is the key to avoiding this situation. (Munindar P. Singh and Michael N. Huhns, 2005)

4.2 Not standardizing SOA

In larger organizations where various IT projects occur concurrently, the need for custom standards is maximum. If different development projects result in the creation of differently designed applications, future integration efforts will be expensive.

The ability for SOA to achieve federation across disparate environments has been well promoted. SOA requires the creation and enforcement of design standards for its benefits to be truly realized.

For example, if one project builds a service-oriented solution in isolation from others, key aspects of its solution will not be in alignment with the neighboring applications it may be required to interoperate with one day.

This can lead to many problems, including:

- Inconsistent data representation that results in disparate schemas representing the same types of information.
- Service descriptions with irregular interface characteristics and semantics.
- Support for different extensions or extensions being implemented in different ways.

SOA promotes a development environment that abstracts back-end processing so that it can execute and evolve independently within each application. However, standardization is still required to ensure consistency in design and interaction of services. (Munindar P. Singh and Michael N. Huhns, 2005)

4.3 Not to have a transition plan

The chances of a successful migration will be severely diminished without the use of a transition plan. Transition plans allow you to coordinate a controlled phasing in service-orientation and SOA characteristics so that the migration can be planned on a technological, architectural, and organizational level.

Examples of typical areas covered by a transition plan include:

- An impact analysis that predicts the extent of change SOA will impose on existing resources, processes, custom standards, and technology.
- The definition of transition architectures, where those environments identified as candidates for a migration to SOA evolve through a series of planned hybrid stages.
- A speculative analysis that takes into account the future growth of Web services and supporting technologies.
- Detailed design changes to centralized logic (such as a new security model).

Creating a transition plan avoids the many problems associated with an ad-hoc adoption of

SOA.

4.4 Not starting with an XML foundation architecture

Since service oriented architecture is text based everything begins with XML. It is this core set of standards that has fueled the creation of the many Web services specifications. It is very impotant that how the data is transported between services that the manner in which this same data is structured and validated behind service lines is often neglected. This oversight can lead to an inconsistent implementation of a persistent XML data representation layer within SOAs.

Standardizing the manner in which core XML technologies are used to represent, validate, and process corporate data as it travels throughout application environments lays the groundwork for a robust, optimized, and interoperable SOA.

4.5 Not figure out SOA performance requirements

When starting out small, it is easy to build service-oriented solutions that function and respond as expected. As the scope increases and more functionality is added, the volume of message-based communication predictably grows. This is when unprepared environments can begin experiencing significant processing latency.

Since SOA introduces layers of data processing, it is subject to the associated performance overhead imposed by these layers. Critical to building a successful service-oriented solution is understanding the performance requirements of your solution and the performance limitations of your infrastructure ahead of time.

This means:

- Testing the message processing capabilities of your environments prior to investing in Web services.
- Stress-testing the vendor supplied processors you are planning to use.
- Exploring alternative processors, accelerators, or other types of supporting technology, if necessary.

Performance is also one of the reasons coarse-grained service interfaces and asynchronous messaging are emphasized when building Web services. These and other design measures can be implemented to avoid potential processing bottlenecks.

4.6 Not understanding Web services security

It is easy to simply continue building on simplistic message exchanges, which usually rely on Secure Sockets Layer (SSL) encryption to implement a familiar measure of security. While SSL can address many immediate security concerns, it is not the technology of choice for SOA. When services begin to take on greater amounts of processing responsibility, the need for message-level security begins to arise. The WS-Security framework establishes an accepted security model supported by a family of specifications that end up infiltrating service-oriented application and enterprise architectures on many levels.

Even if your vendor platform does not yet provide adequate support for WS-Security, and even if your current SSL-based implementation is meeting immediate requirements, it is also advisable to pay close attention to the changes that are ahead. Proceeding without taking WS-Security into account will inevitably lead to expensive retrofitting and redevelopment.

4.7 Not keeping in touch developing technology

This is the result of establishing a vendor-neutral communications framework that allows solutions based on disparate technologies to become fully interoperable.

5. USAGE OF SOA

In practice there are many different usage areas of SOA. The reason of using SOA in following applications is taking the advantages of SOA.

5.1 In e-education systems

In today’s technology designing and implementing large systems is very difficult. Because large systems have a complex and decentralized structure. Service oriented architecture is a new technology for distributed systems. It facilitates the development of such systems by supporting modular design, application integration and interoperation, and software reuse. As a result of these properties it decreases the cost and effort needed for implementation and design. With open standards, such as XML, SOAP, WSDL and UDDI, the Service Oriented Architecture supports interoperability between services operating on different platforms and between applications implemented in different programming languages. (Parichat Pasatcha and Komrhon Sunat, 2008)

In Figure 4.1, the general structure of the e-education system can be seen.

Client: Various types of computer devices are present in the client that can use diverse platforms.

Education Server: The education server of this e-education architecture is composed of several components that include web service, middleware, database, e-learning services and educational services.

Service Provider: A service provider makes the services of one organization available to others in a controlled and secure manner.

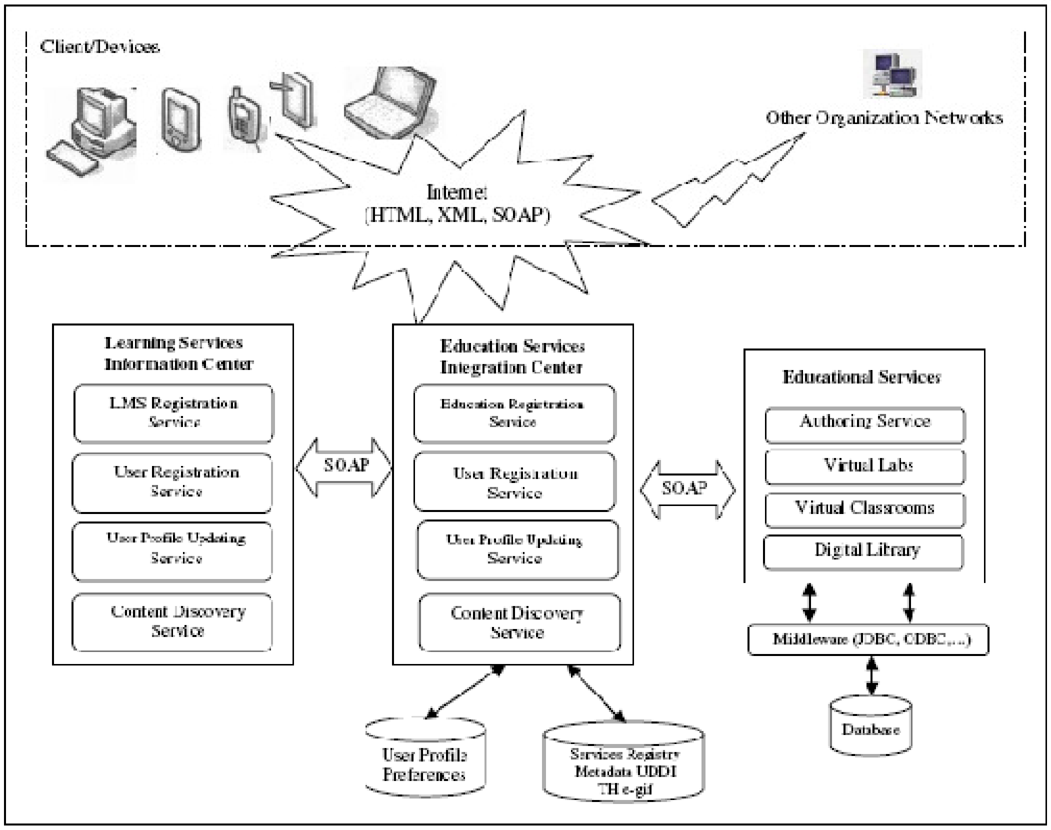

Client/Devices
Other Organization Networks
Internet (HTML, XML, SOAP)
Learning Services Information Center
LMS Registration Service
User Registration Service
User Profile Updating Service
Content Discovery Service
SOAP
Education Services Integration Center
Education Registration Service
User Registration Service
User Profile Updating Service
Content Discovery Service
SOAP
Educational Services
Authoring Service
Virtual Labs
Virtual Classrooms
Digital Library
Middleware (JDBC, ODBC,...)
Database
User Profile Preferences
Services Registry Metadata UDDI TH e-gif

Figure 4.1 High-level view of e-education system architecture.
(Parichat Pasatcha and Komrhon Sunat, 2008)

Education Service Integration Center: Is a local SCORM compatible LMS and serves as a learning service broker which obtains the user requests from web client components and subsequently invokes the corresponding services to serve the user requests.

Service Requestor: The interface, binding, and service endpoint of a education service are defined in WSDL files. The Requestor must support the WS-Inspection specification, and the service requesters can easily locate the WSDL documents of the Web services that are exposed by the Router. The users uses the UDDI registry to obtain WSDL interface definition and implementation definitions of the learning services in virtual organization.

Service Provider: It provides a taxonomy and details of available services to systems that participate in an SOA. The learning service directory might be an open-standard UDDI directory and TH e-gif (Thailand e-Government Interoperability Framework), or more basic forms might be implemented as a design-time service catalogue, perhaps using collaboration technology.

5.2 In military communication networks

The principles of network enabled capability highlight the need for seamless information exchange. The service oriented architecture has been recognized as one of the key enablers to achieve this. At the same time, Web services have become a standard for implementing service oriented architecture. However, these technologies have been developed for environments with abundant data rates, environments which are very different from military tactical networks.

This system requires enabling data exchange at al operational levels. Radio systems such as HF or VHF may have a data transfer rate lower than 1 Kb/s in practice, due to the need for long range signals and jamming resistance. It is thougt that there are measures that can be taken to use SOA technologies over disadvantaged grids.

But using web serivces for this purpose have some this advantages. One of them is XML documents that has large sizes. Because there is a limited network capacity. If the size of message is very large it will cause a big latency in the system. To pass over this problem it is decided to reduce the size of the XML messages. The best way to make this is compression. The size of the XML messages will be reduces by general compression techniques such az GZIP or binary XML. General compression techniques, such as GZIP, can compress any kind of data, both XML and non-XML; binary XML operates only on XML data. Using binary XML is more useful. In general compression techniques the structure of the XML is lost. So end points must unzip and than get a good structured XML. In binary XML the structure of the XML is not lost, so it decreases the processing time. The following graph shows us the performance of different compression techniques. (Ketil Lund et al., 2007)

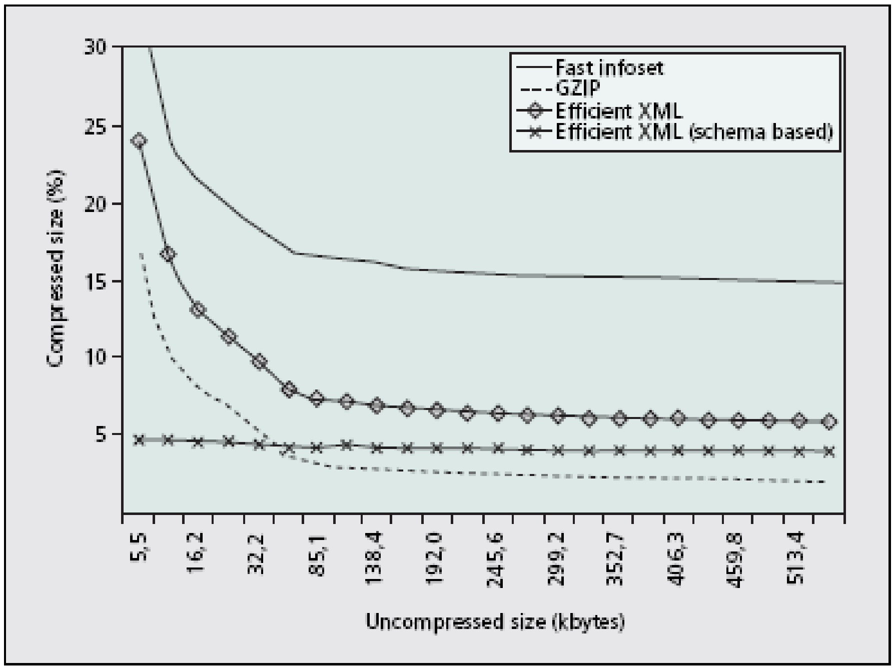


Figure 4.2 Results of different compression techniques.
(Ketil Lund et al., 2007)

After an efficient data representation is established, the next step is to ensure efficient data exchange. The main point here is to take advantage of the properties of the underlying network to achieve a more efficient data transfer.

Three different mechanisms for information exchange are described.

• *Referentially complete message information exchange* means exchange of self-contained XML documents. Using Web services, this mechanism can be realized using either a traditional request-response mechanism or a publish/subscribe mechanism.

• *Replication-based information exchange* implies that the participants first perform an initial exchange of data in order to start off with identical data sets. Subsequently, only updates are exchanged between the parties using push technology, that is, publish/ subscribe when using Web services.

• *Query based information exchange* can be compared to traditional request-response information exchange and is also realized this way in Web services. One party requests information from another by submitting a query, which in turn returns the result of this query. (M. Gerz et al.,2006)

To promote the NEC vision, the SOA services must be made available to users at all operational levels. One of the challenges in this context is to use Web services over the heterogeneity of communication systems used in the

different operational levels in a military network. Since HTTP is synchronous it will lose the connection in time delays, so using HTTP for communication will not be a good solution. A better solution is a store-and-forward overlay network that can cope with the differences of military communication networks. Such an overlay network must be asynchronous and message-based, because it will be necessary to store and forward the information in transit between the networks. As a result of this another protocol that is called MMHS is used.

5.3 SOA with grid computing

Service-based approaches are rising to prominence because of their potential to meet the requirements for distributed application development in e-business and e-science. The emergence of a service-oriented view of hardware and software resources raises the question as to how database management systems and technologies can best be deployed or adapted for use in such an environment.

The principal strengths of WSs and Grid middlewares are complementary, with WSs focusing on platform-neutral description, discovery and invocation, and Grid middlewares focusing on the dynamic discovery and efficient use of distributed computational resources.

In this research a distrubuted query processing system is designedand it is called OGSA-DQP. This system:

- Supports low-cost data integration
- Builds on parallel database technology

The importance of DQP in a service-based Grid setting provides a claim of mutual benefit: the Grid stands to benefit from DQP, through the provision of facilities for declarative request formulation that complement existing approaches to service orchestration; and DQP stands to benefit from the Grid, due to the support provided for the discovery and allocation of computational resources, as required to support computationally demanding database operations, and implicit parallelism for complex analyses.

In a service-oriented Grid the principal objective is to enable computational resources to be accessed and managed in a secure and systematic manner. OGSA Data Access and Integration (OGSA-DAI) project has developed a service-based infrastructure for accessing both relational databases and XML repositories .

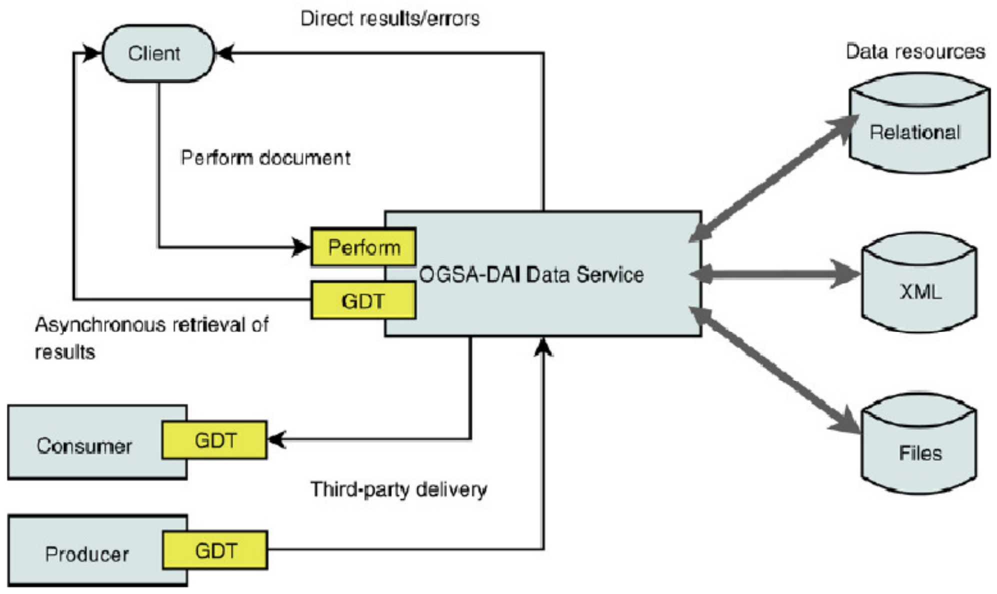


Figure 4.3 OGSA-DAI data interface
(Steven Lynden et al., 2008)

•OGSA-DAI data service: A WS that implements various port types allowing the submission of requests and data transport operations.

•Client: An entity that submits a request to the OGSA-DAI data service; a request is in the form of a perform document that describes one or more activities to be carried out by the service.

•Consumer: A process, other than the client, to which an OGSADAI service delivers data.

•Producer: A process, other than the client, that sends data to an OGSA-DAI data service.

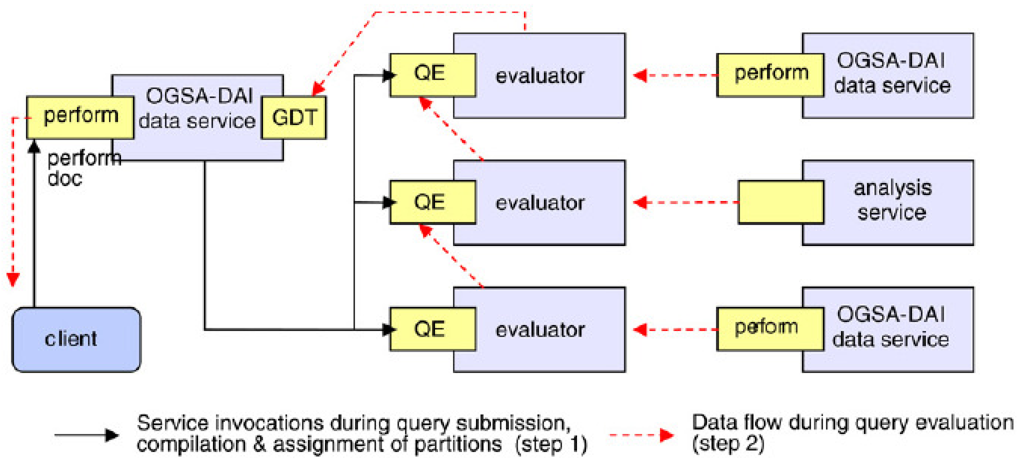


Figure 4.4 OGSA-DQP architecture
(Steven Lynden et al., 2008)

OGSA-DQP extends OGSA-DAI with the following services:

•DQP coordinator service: An OGSA-DAI data service, enhanced to support distributed queries using the extensibility points discussed in the previous section. The main enhancement is the contribution of the DQPQueryStatement activity which compiles, optimises and schedules SQL queries for execution.

•DQP evaluator service: A service capable of evaluating a query fragment provided by the coordinator. An evaluator is able to play a role in the evaluation of a query by retrieving data from OGSA-DAI-wrapped data resources, invoking analysis services and managing the flow of data between other evaluators. Multiple evaluators are used, to provide the benefits of parallelism during query evaluation. (Steven Lynden et al., 2008)

As it is shown in the figure the response time decreases if the number of parallen working services increase.

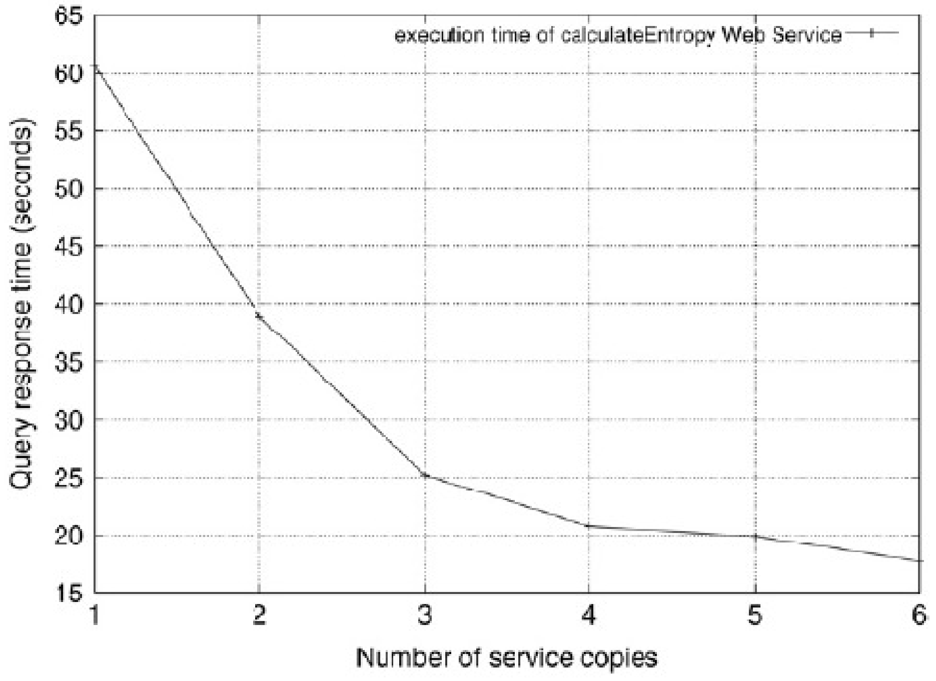


Figure 4.5 Response times for analysis queries with different levels of parallelism (Steven Lynden et al., 2008)

These usage areas are not only the areas that we can use service oriented architecture. There are many different applications that are implemented by using service oriented architecture.

5.4 SOA Solution Providers

SOA can be implemented using kind of suites like Microsoft Biztalk Server, TIBCO suite, IBM websphere, SUN J2EE, Oracle SOA Suite etc.

5.4.1 Microsoft BizTalk Server

Microsoft BizTalk Server, often referred to as simply "BizTalk", is a business process management (BPM) server. Through the use of "adapters" which are tailored to communicate with different software systems used in a large enterprise, it enables companies to automate and integrate business processes. Offered by Microsoft, it provides the following functions: Business Process Automation, Business Process Modeling, Business-to-business Communication, Enterprise Application Integration and Message broker.

In a common scenario, BizTalk enables companies to integrate and manage business processes by exchanging business documents such as purchase orders and invoices between disparate applications, within or across organizational boundaries.

Development for BizTalk Server is done through Visual Studio .NET. (http://msdn.microsoft.com/en-us/library/aa562161.aspx, 20 January 2009)

5.4.2 IBM WebSphere

IBM WebSphere Application Server drives business agility by providing millions of developers and IT Architects with an innovative, performance based foundation to build, reuse, run, integrate and manage Service Oriented Architecture (SOA) applications and services. It can run applications and services in a reliable, highly available, secure and scalable environment to ensure business opportunities are not lost due to application downtime. Runtime performance through provisioning optimized, web services, and EJB3 enhancements which can result in fewer energy consuming processors performing the same workloads of previous versions. New security capabilities add deeper levels of management, user governance and auditing to decrease system vulnerabilities while maximizing developer productivity.

From service-enabling legacy assets to inventing new ones, our technology makes your business accessible to new users in innovative ways, it gives immediate insight and interaction with partners, suppliers and customers and increasing your return on investment. WebSphere Application Server V7 supports the broadest range of platforms in the industry, helping provide assurance that your applications can be built to run on the platform that most makes sense for business applications. Increase developer productivity using enhanced support for standards, emerging technology and a choice of development frameworks that simplify programming models. Operating systems supported: AIX, HP Unix,ifamily,Linux,SunSolaris,Windows,z/OS(IBM, http://www-01.ibm.com/software/webservers/appserv/was/, 20 January 2009)

5.4.3 TIBCO

TIBCO is doing with ActiveMatrix is shifting beyond its traditional integration focus and providing a rear container for the development and deployment of services.TIBCO extends to service bus in two ways. One is into the tooling, from a developer perspective, they're abstracting a lot of the glop they need to tie code into an ESB, and TIBCO is trying to do something similar to that.

It's much more declarative. It's all about annotations and policies to be attached to things, rather than code written. On the other side, TIBCO are unlike a lot of the other ESB players. They are trying to natively support .NET, so they actually have a .NET container that you can write .NET components in and hook them into the service bus natively. This specification might not seen from anywhere else, apart from Microsoft. There's two ways in which they're moving beyond the basic ESB proposition.

It's much more of a development infrastructure focus than an integration infrastructure focus. ActiveMatrix appears to be something with JBI, a Java Business Integration implementation, basically a kind of standards-based plug-and-play ESB on steroids.

There are loads of tools around to help developers take existing Java code, or whatever, right-click on it, and create SOAP and WSDL bindings, and so on. But, there are other issues of quality, consistency of interface definitions, and use of schemas — more leading-edge thinking around using policies, for example. This would involve using policies at design time, and then having those enforced in the runtime infrastructure to do things like manage security automatically and help to manage performance, availability, and so on. (TIBCO ,www.tibco.com, 20 january 2009)

5.4.4 SUN J2EE

Sun's Java Web Services Developer Pack 1.5 (Java WSDP 1.5) and Java 2 Platform, Enterprise Edition (J2EE) 1.4 can be used to develop state-of-the-art web services to implement SOA. The J2EE 1.4 platform enables you to build and deploy web services in your IT infrastructure on the application server platform. It provides the tools you need to quickly build, test, and deploy web services and clients that interoperate with other web services and clients running on Java-based or non-Java-based platforms. In addition, it enables businesses to expose their existing J2EE applications as web services. Servlets and Enterprise JavaBeans components (EJBs) can be exposed as web services that can be accessed by Java-based or non-Java-based web service clients. J2EE applications can act as web service clients themselves, and they can communicate with other web services, regardless of how they are implemented. (JAVA SUN SOA, http://java.sun.com/developer/technicalArticles/WebServices/soa/, 20 january 2009)

5.4.5 ORACLE SOA SUITE

This tool is used in implementation section that is explained in detail below chapter but in brief explanation Oracle Soa Suite: provides a flexible and dynamic IT Infrastructure that transforms brittle systems, applications and data sources into highly flexible, reusable service components. The resources offered here cover a wide range of integration topics, including business process management (BPM), the service bus, and data services. (OTN, http://www.oracle.com/technology/architect/soa/soaint/index.html, 21 January 2009)

6. ORACLE SOA

This final chapter includes the details of the used SOA framework and the designed/developed sample application. Thus this final chapter is started with Oracle SOA Suite that has been used to address the concepts mentioned in the previous chapters. Later on, a basic sample application has been told and shown (using snapshots); how to develop/deploy/run/test the application from scratch.

6.1 Description of Oracle SOA

Before talking about the Oracle SOA suite, a short summary of SOA technologies / terminologies / philosophy ought to be given; SOA provides an enterprise architecture that supports building connected enterprise applications. SOA facilitates the development of enterprise applications as modular business Web services that can be easily integrated and reused, creating a truly flexible, adaptable IT infrastructure.

Oracle SOA Suite (in which this chapter is totally based on) is a complete set of service infrastructure components for creating, deploying, and managing business services. Oracle SOA Suite enables services to be created, managed, and orchestrated into composite applications and business processes. In addition, it is possible to benefit from the common security, management, deployment architecture, and development tools that are provided as out-of-box.

Oracle SOA Suite is a standards-based *best-of-breed* technology suite that consists of the following:

- Integrated Service Environment (ISE) to develop services
- Oracle BPEL Process Manager to orchestrate services into business processes
- ESB to connect existing IT systems and business partners as a set of services
- Oracle Business Rules for dynamic decisions at runtime that can be managed by business users or business analysts
- OracleAS Integration Business Activity Monitoring to monitor services and disparate events and provide real-time visibility into the state of the enterprise, business processes, people, and systems.
- Oracle Web Services Manager to secure and manage authentication, authorization, and encryption policies on services that is separate from application service logic

- UDDI registry to discover and manage the lifecycle of Web services.
- Oracle Application Server 10g Release 3 (10.1.3) to provide a complete Java 2, Enterprise Edition (J2EE) 1.4-compliant environment for J2EE applications.

Figure 6.1 depicts the Oracle SOA suite and its components briefly. Components could be listed as;

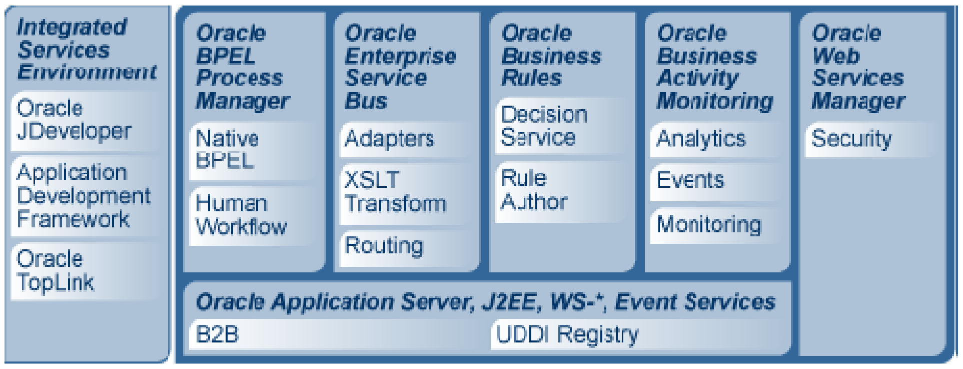

Integrated Services Environment
Oracle JDeveloper
Application Development Framework
Oracle TopLink
Oracle BPEL Process Manager
Native BPEL
Human Workflow
Oracle Enterprise Service Bus
Adapters
XSLT Transform
Routing
Oracle Business Rules
Decision Service
Rule Author
Oracle Business Activity Monitoring
Analytics
Events
Monitoring
Oracle Web Services Manager
Security
Oracle Application Server, J2EE, WS-*, Event Services
B2B
UDDI Registry

Figure 6.1 Oracle SOA Suite Architecture(Oracle Soa, 15 October 2008)

6.1.1 Integrated Service Environment

Oracle JDeveloper (ORACLE ISE, 15 October 2008) is the development component of Oracle SOA Suite. It forms a comprehensive ISE for developing, composing, and orchestrating services into business processes. Business processes can be deployed, registered, and consumed from several types of user interfaces, including desktop clients, browsers, and mobile and telnet devices.

JDeveloper enables developers to model, create, discover, assemble, orchestrate, test, deploy, and maintain composite applications based on services. JDeveloper supports SOA principles and XML Web services standards, as well as traditional Java, J2EE, and PL/SQL component and modular code mechanisms.

6.1.2 Oracle BPEL Process Manager

Oracle BPEL Process Manager (ORACLE BPEL, 15 October 2008) provides a framework for easily designing, deploying, monitoring, and administering processes based on BPEL standards. Oracle BPEL Process Manager provides support for the following features:

- Web service standards such as XML, SOAP, and WSDL

- Dehydration (enables the states of long-running processes to be automatically maintained in a database) and correlation of asynchronous messages
- Parallel processing of tasks
- Fault handling and exception management during both design time and run time
- Event timeouts and notifications
- Compensation mechanisms for the implementation of long-running transactions
- Scalability and reliability of processes
- Management and administration of processes
- Version control
- Audit trails for tracing business flow history

Oracle BPEL Process Manager adds value and ease of use to BPEL functionality by providing support for the following in the JDeveloper BPEL Designer:

- Transformations, workflows, worklists, notifications, and sensors
- Technology adapters, including file, FTP, database, advanced queuing (AQ), Java Messaging Service (JMS), Oracle Applications for Oracle E-Business Suite, and WebSphere MQ
- Third-party adapters, including J.D. Edwards OneWorld, PeopleSoft, SAP R/3, Siebel, Tuxedo, CICS, VSAM, IMS/TM, and IMS/DB

6.1.3 Oracle Enterprise Service Bus (ESB)

An enterprise service bus (ORACLE ESB, 15 October 2008) moves data among multiple endpoints, both within and outside of an enterprise. It uses open standards to connect, transform, and route business documents (XML messages), among disparate applications. It enables monitoring and management of business data, with minimal impact on existing applications.

6.1.4 Oracle Business Rules

Oracle Business Rules (ORACLE BR, 16 October 2008) enables dynamic decisions at runtime allowing, among other features, applications to rapidly adapt to regulatory and competitive pressures. This increased agility can be used by developers and analysts at the same time without stopping the business flows.

6.1.5 OracleAS Integration Business Activity Monitoring

OracleAS Integration Business Activity Monitoring (ORACLE BAM, 16 October 2008) gives business executives the ability to monitor their enterprise business services in real-time and to correlate their KPIs (key performance indicators) to the actual business process. Thus, Oracle BAM is a complete solution for building real-time operational dashboards, monitoring and alerting applications.

6.1.6 Oracle Web Services Manager

Oracle Web Services Manager (ORACLE WSM, 16 October 2008) is a security administrator's environment designed to secure access to Web services and monitor activities performed on protected Web services.

6.1.7 OracleAS UDDI Registry

OracleAS UDDI Registry (ORACLE UDDI, 16 October 2008) provides a key component of any SOA with a configurable, scalable, secure repository of Web services that can be managed, discovered and governed by Oracle Fusion Middleware.

6.1.8 Oracle Application Server

Oracle Application Server (ORACLE AS, 16 October 2008) is a standards-based application server that provides a comprehensive and fully integrated platform for running Web sites, J2EE applications, and Web services.

7. SOA Design & Implementation of Online Campus

I have implemented an Online Campus management system using the principles of the Service Oriented Architecture using Oracle SOA Suite and Microsoft .NET environment.

Many campus management solutions are usually offline and closed systems. A university campus consists of many different parts such as registrars, librarians, financial department etc… These different parts use different packet programs to do their jobs. However they process the data on the same students so they usually have to interact with themselves. This is not possible by using closed systems. Instead, a campus management system should be open and all of the different campus parts should be able interact with themselves freely on such systems. This idea is very suitable with Service Oriented Architecture.

In my design, all the different departments of the campus deploy their specific services on Oracle SOA Suite. So, all of these services can be consumed by anyone on the campus. For example a librarian can get information about a student by using registrar's service about the student information. As an another example, a registrar can have a lookup on the librarian if a student has a unreturned book so that he/she can hold the student from registering courses.

This design also increases the security of the campus. Since the system is divided into many layers, an end user can not directly access to the core of the system, the database. The user interface, which is the campus website, can only interact with the Oracle SOA Suite. It has a very strong security system whose boundaries are hard to pass by hackers.

The database of the Online Campus is implemented in Microsoft SQL Server. It is designed for fast and reliable access thorugh relational integrity and indexes. The code access to the database is done with Microsoft's last technology called SQL to Linq. Linq (Language Integrated Query) creates an automatic database to object mapping of the Online Campus database and in the end it creates classes of all tables and its relations. Querying is done through the Linq syntax, which is the latest addition to C# language. No single classic SQL query was written. Linq also handles the security for SQL injections so database protection from SQL injection is secured. Figure 7.1 is the database diagram of Online Campus.

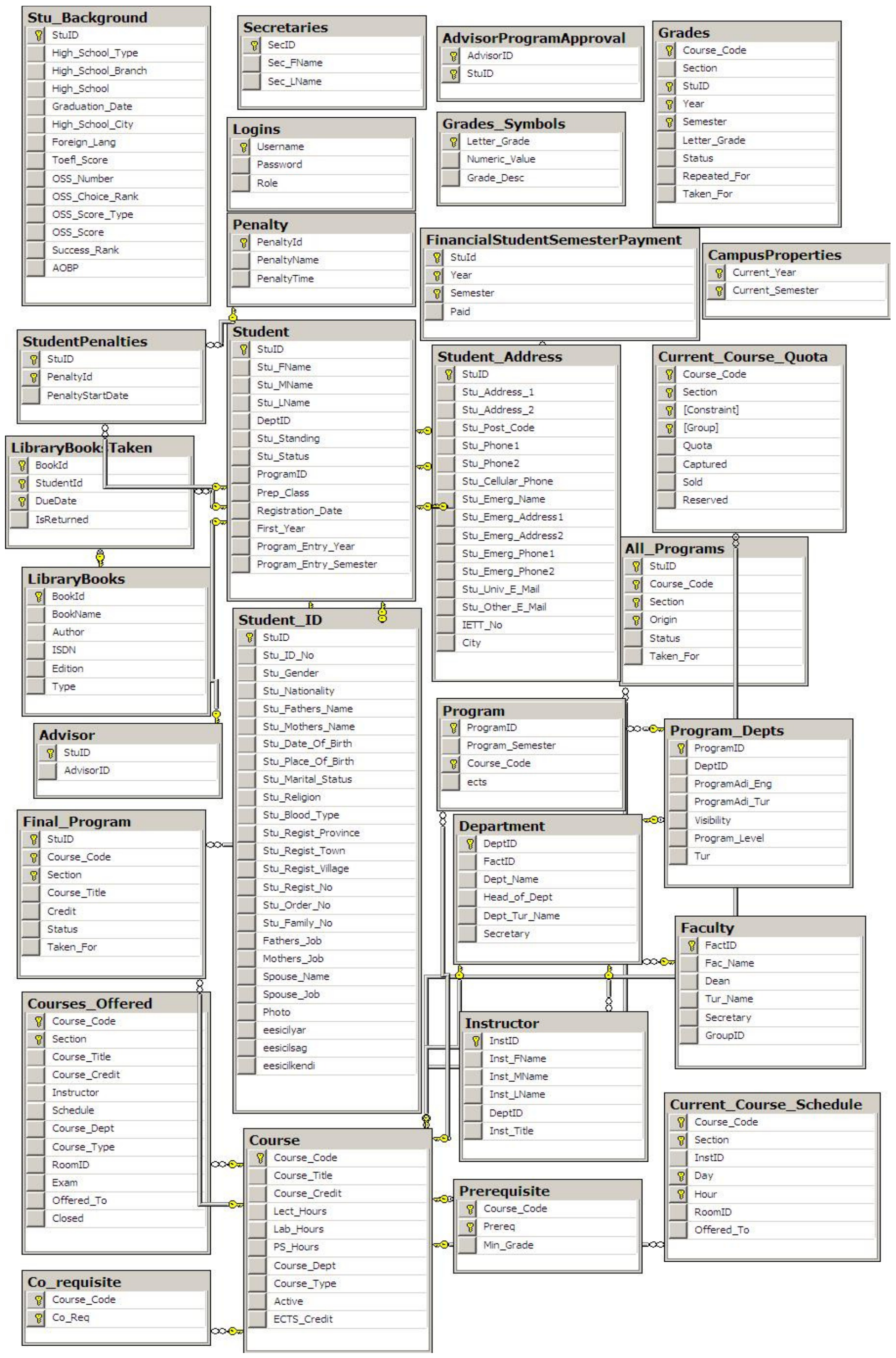


Figure 7.1. Online Campus Database

I implemented 50 web services for the Online Campus. In the design process, I chose the most common webservices that could be consumed by anybody and left the details to the frontend implementation. The list of the webservices is below. Now, I will explain some of the services in the proceeding sections.

7.1 Web Services

7.1.1 GetLoginAccount

Gets the details of the given account id.

7.1.2 GetStudentGrades

This service returnsall grades that the given student has.

7.1.3 GetStudentSelectedCourses

Returns the courses that the given student selected during the courses registration process.

7.1.4 GetStudentSelectedCoursesNumber

This service give how many courses the given student registered during registration. Used by the advisor.

7.1.5 GetStudentSelectedCoursesNumberOfConflicts

Returns the conflicted courses number during the registration.

7.1.6 GetStudentSelectedCoursesTotalCredit

Returns the amount of credit the student registered in the current semester.

7.1.7 GetStudentsWaitingForApproval

Returns the students that are waiting for approval from the given advisor.

7.1.8 GradeTheStudent

Gives grade to the given student fort he given course and section.

7.1.9 IsStudentProgramApproved

Checks if the given student's program had been approved by its advisor.

7.1.10 LendBookToStudent

The librarian lends the given book the given student.

7.1.11 ReturnBook

The librarian returns the given book from the given student.

7.1.12 UpdateBook

Librarian updates the given book with the given information.

7.1.13 UpdateCourse

This service updates the given course with the given informationç

7.1.14 UpdateDepartment

The given department is updated with the given information.

7.1.15 UpdateInstructor

The given instructor is updated with the given information.

7.1.16 UpdateLogin

The given credential is updated with the given information.

7.1.17 UpdateStudent

The given student is updated with the given information.

7.1.18 GetStudentGPA

Calculates and return the given student's overall GPA

7.1.19 AddStudentFinancialHold

The financial staff hold the student a financial hold.

7.1.20 RemoveFinancialHold

The financial staff removes the hold from the given student.

7.1.21 GetStudentSemesterGPA

Returns the given semester's GPA fort he given student.

7.1.22 IsStudenEligibleForRegistration Service

This service is to determine if a given student can register courses. First, it calls the registrar service to see if the student has any penalty. If the student has any penalty it replies to the client in negative way. If the student doesn't have any penalty, it calls the library service to see if the student has any unreturned books. If the student has any unreturned book, it replies negatively to the client, other wise it continues calling the financial department's service to check if the student has any unpaid tuition. If the student has any payment, it replies negatively, otherwise it returns positively.

This service is used in registration period. If the service gives any negative reply, the student can not proceed to the registration. Below are the bpel process designs of the service in Oracle SOA Suite.

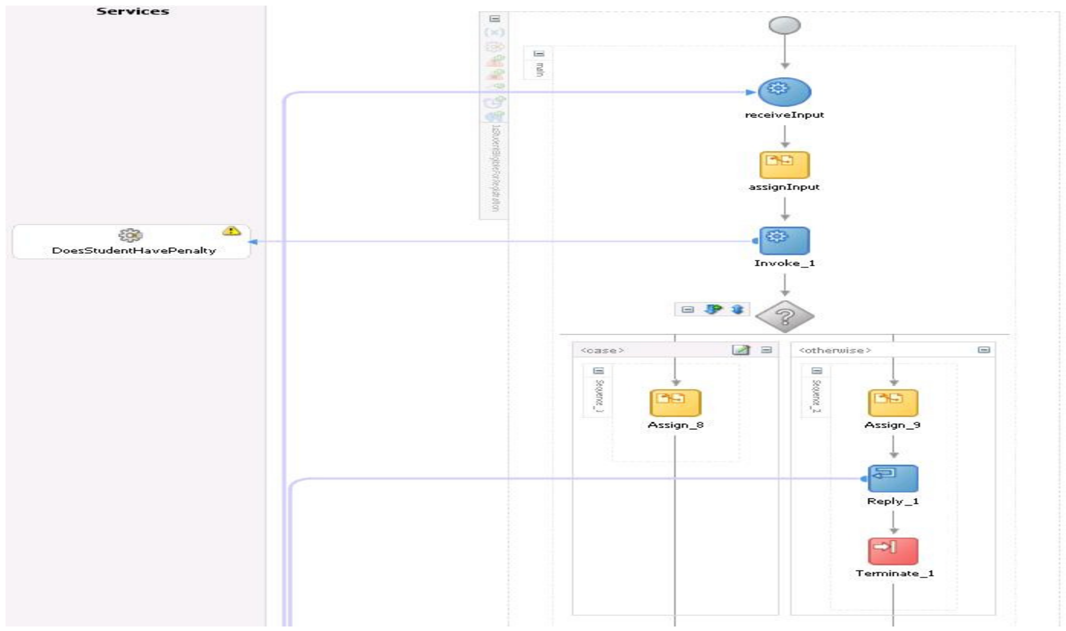


Figure 7.2.a IsStudentEligibleForRegistration

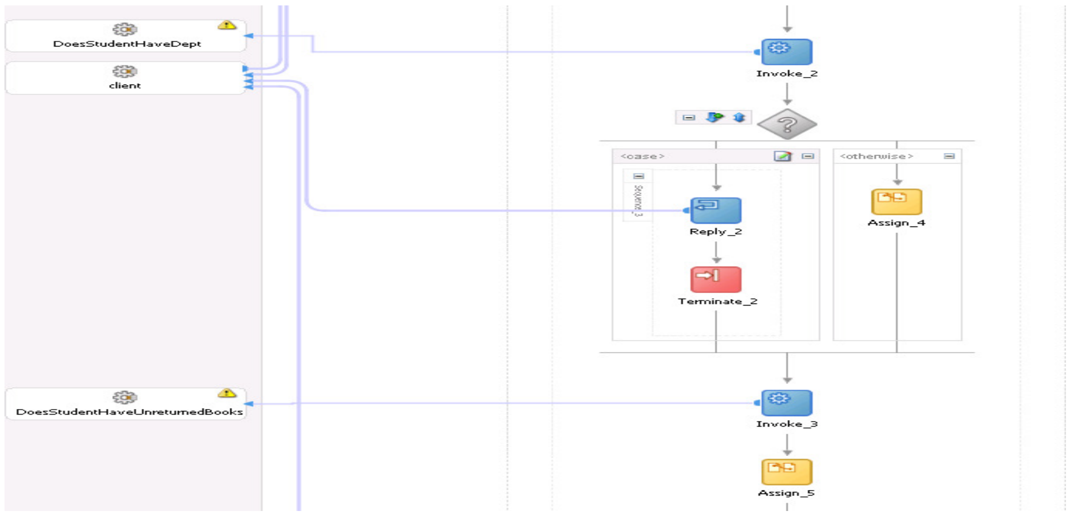


Figure 7.2.b IsStudentEligibleForRegistration

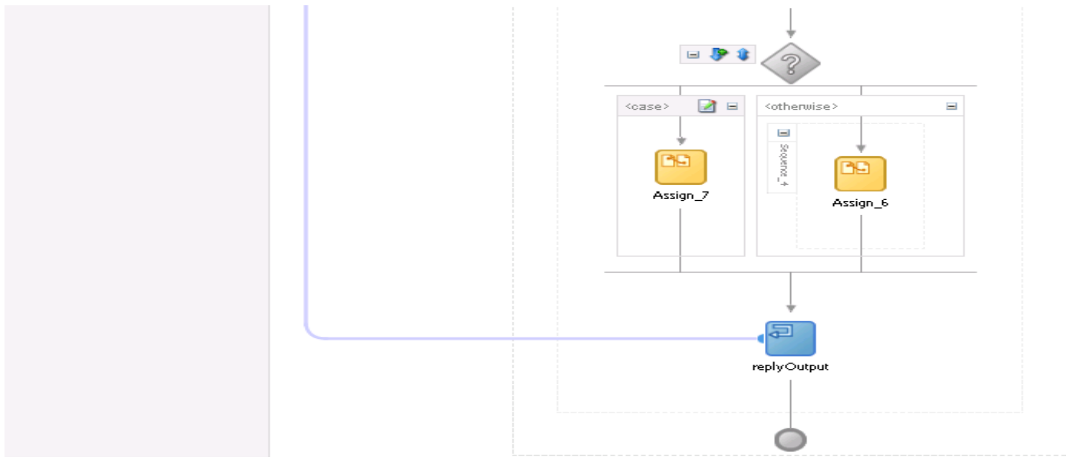


Figure 7.2.c IsStudentEligibleForRegistration

7.1.23 GetStudentSubmittedCoursesInformation Service

This service gathers information about the student registration and returns it to the client. It first gets the number of courses the student registered. Then it gets the number of the total credits that student took. Lastly it gets the number of conflicted classes of the registered courses. Then the service assigns these values in a final data and returns it the client.Figure 7.2 shows the BPEL process design of the service.

7.1.24 GetStudentInformation Service

This service gets all the information from the registrar web service and returns it to the client. The assignment of the student data is done through the SOA Oracle Suite assignment system. Figure 7.3 shows the BPEL design of the service.

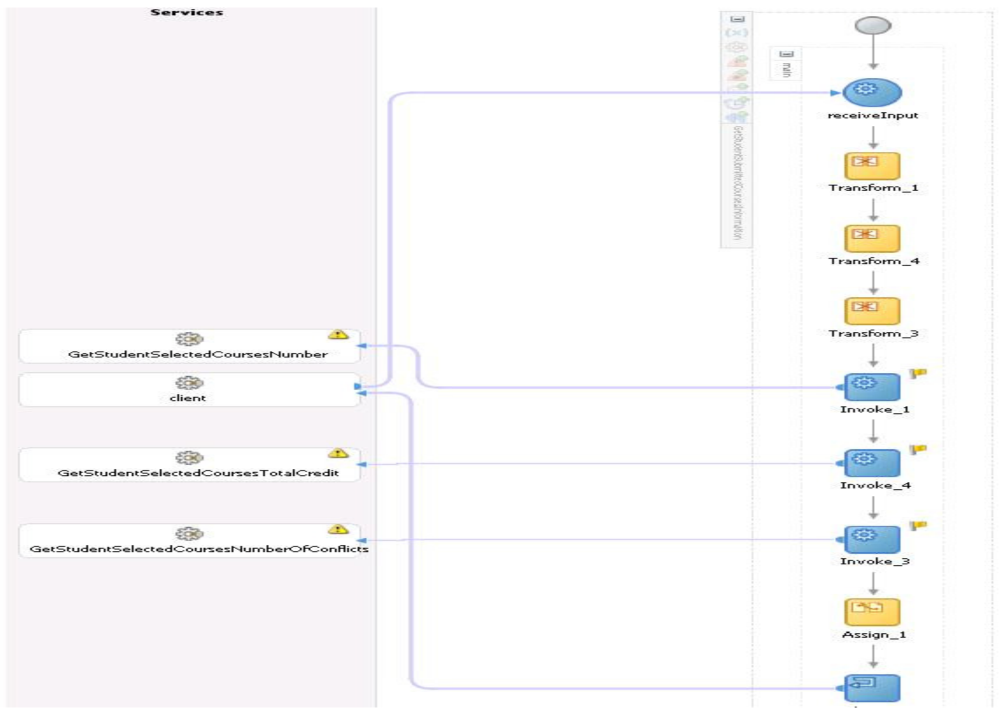

Services
GetStudentSelectedCoursesNumber
client
GetStudentSelectedCoursesTotalCredit
GetStudentSelectedCoursesNumberOfConflicts
main
GetStudentSubmittedCourseInformation
receiveInput
Transform_1
Transform_4
Transform_3
Invoke_1
Invoke_4
Invoke_3
Assign_1

Figure 7.3. GetStudentSubmittedCourseInformation BPEL Process design

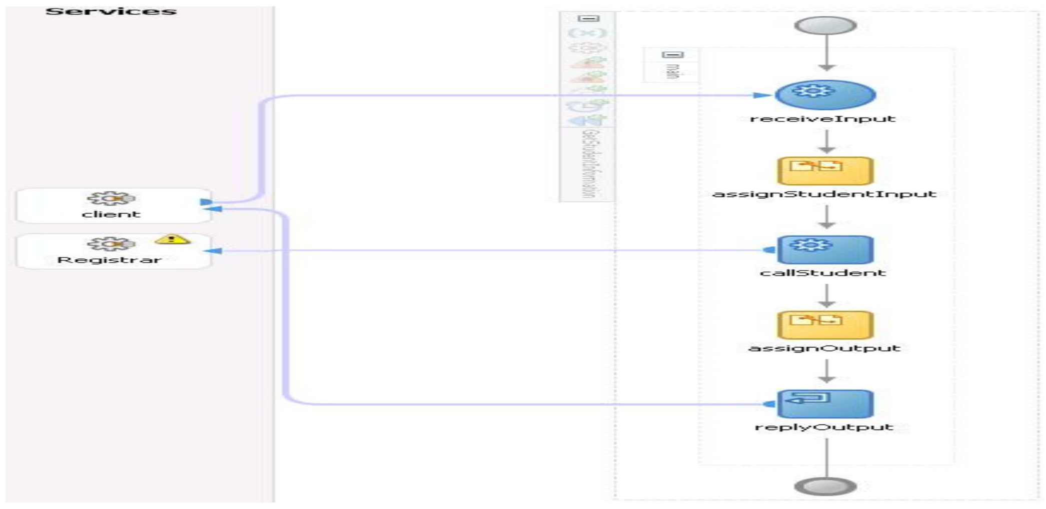


Figure 7.4. GetStudentInformation BPEL process design

7.1.25 GetStudentAddressInformation Service

This service gets the student's address information from the registrar eService and returns it to the student. The student address data is assigned through the Oracle SOA Suite assignment system.

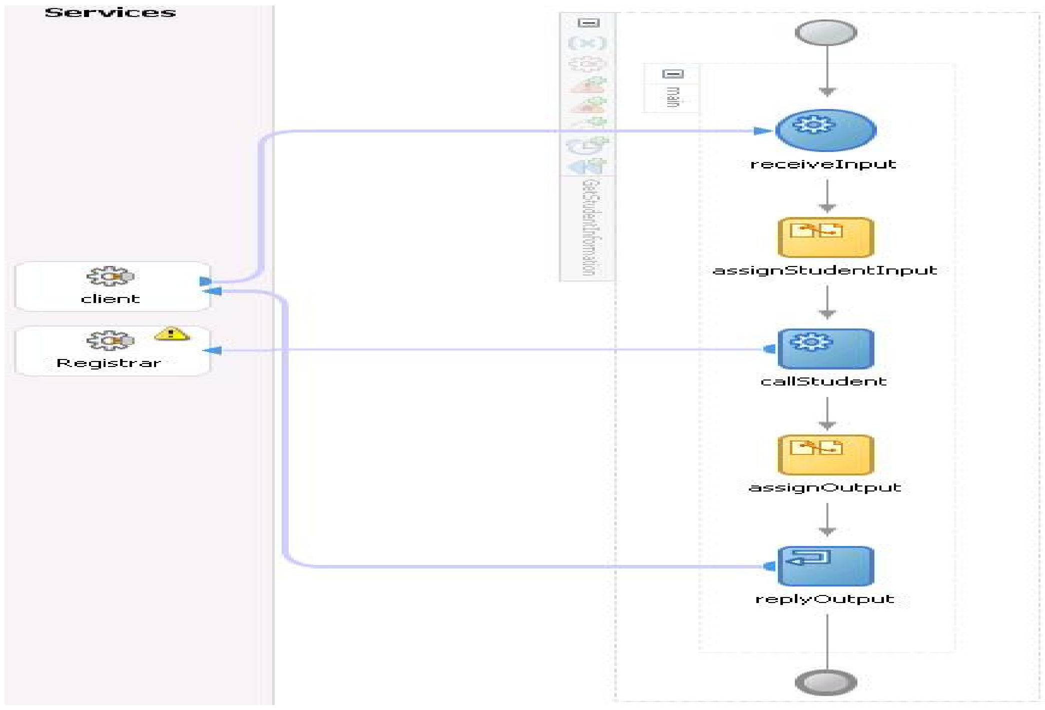


Figure 7.5. GetStudentAddressInformation BPEL Process design

7.1.26 AddStudentAddressInformation Service

This service gets the address information of the student from the client and passes it to the registrar's student addition web service. This service uses xml transformation language (XSLT) as the data assignment system.

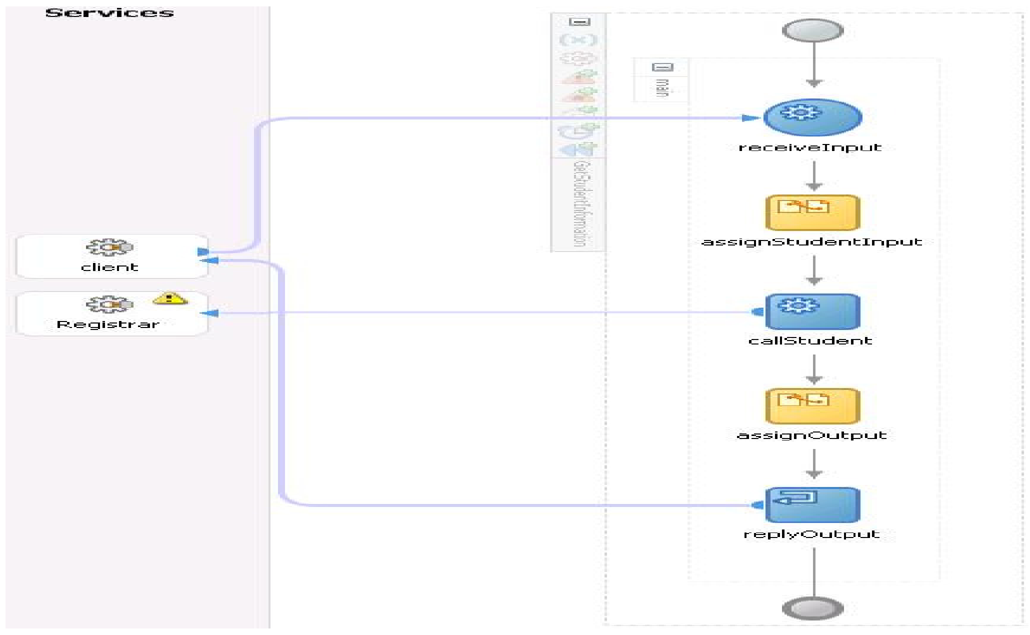


Figure 7.6. AddStudentAddressInformation BPEL Process design

7.1.27 GetCourseInformation Service

This service gets the information about a course and returns it to the user. Course information includes the course code, course title, the schedule of the course, the instructor who teaches etc...

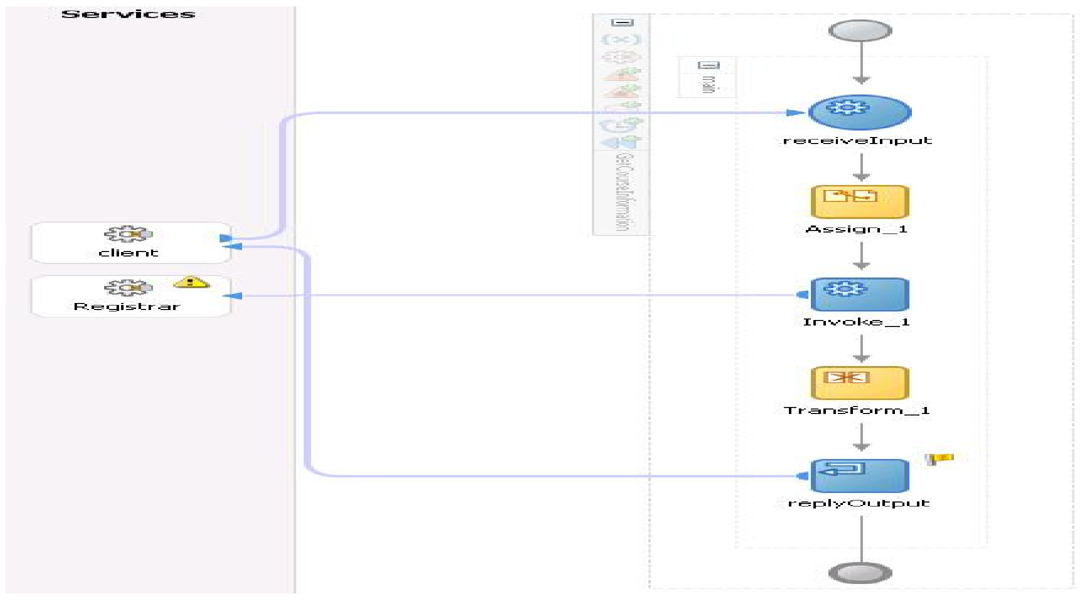


Figure 7.7. GetCourseInformation BPEL Process Design

7.1.28 GetStudentSelected Courses Service

This service retrieves the given student’s selected courses for the registration. Note that, these courses are only selected courses not registered courses. These may be used to advisor to approve or student can work again on these courses for selection.

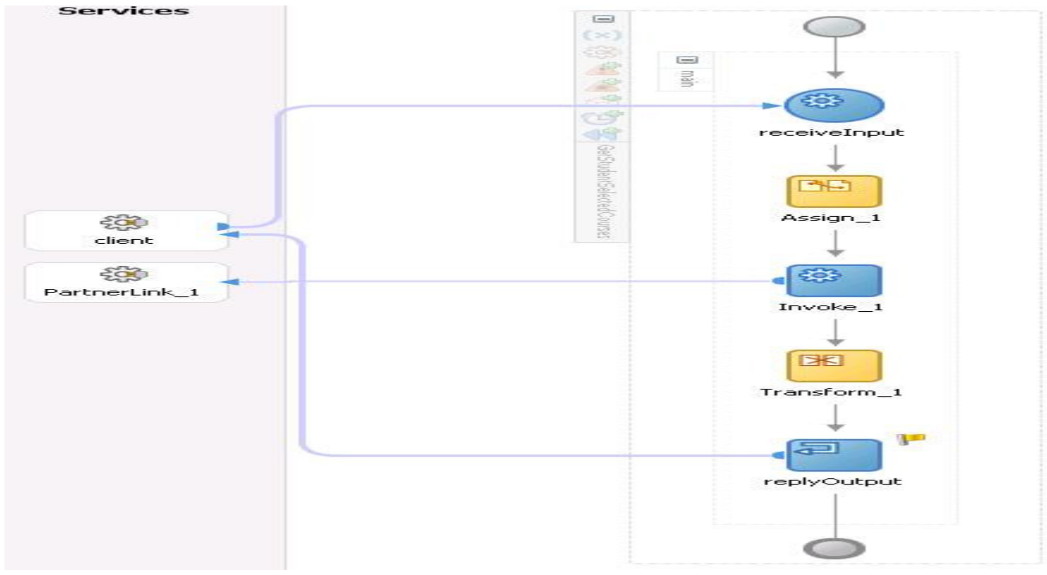


Figure 7.8. GetStudentSelectedCourses BPEL Process design

7.1.29 GetInstructor Service

This service retrieves information about the instructor and returns it to the client. The information includes the id, name, last name title of the instructor.

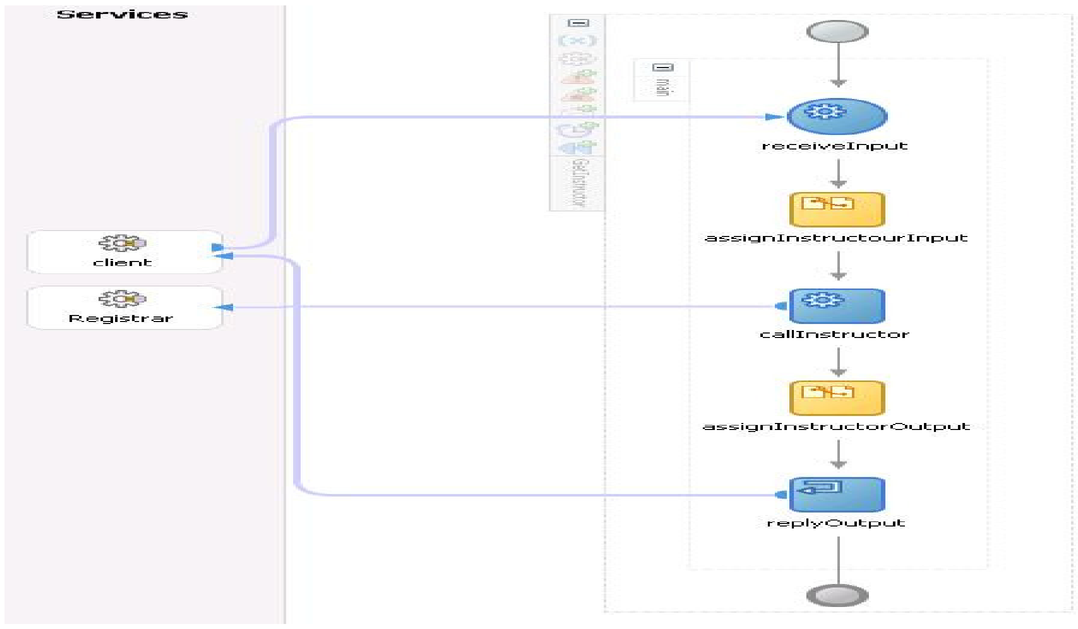


Figure 7.9. GetInstructor BPEL Process design

7.1.30 DisapproveStudentProgram Service

This service is to disapprove the student registration process. If the advisor is not happy with the student's program, he/she can disapprove it with the service.

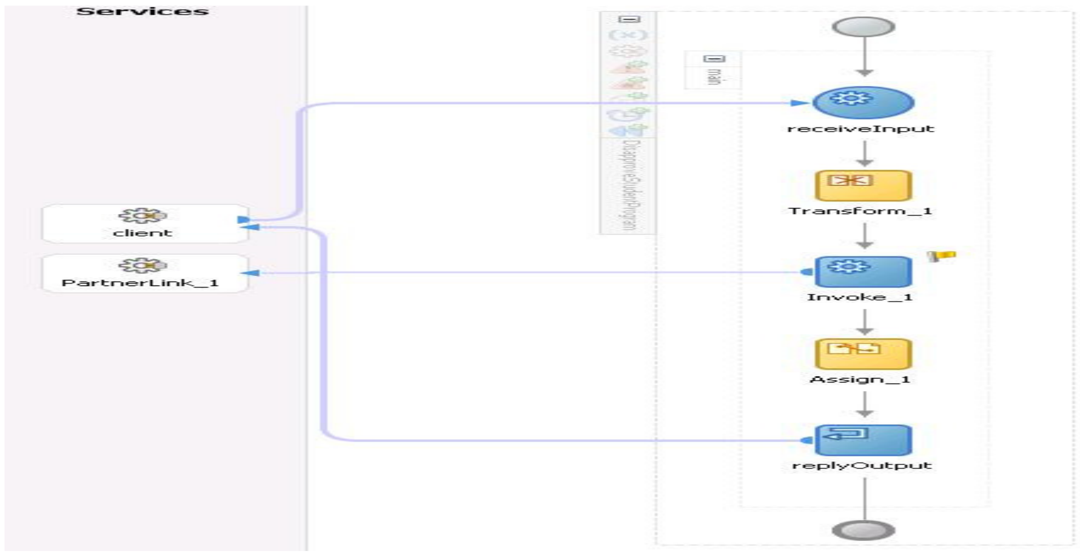


Figure 7.10. DisapproveStudentProgram BPEL design

7.1.31 GetCoursesOffered Service

This service gets the current semester's offered courses from registrar web service. When it gets the data, it transforms it with XSLT. Figure 7.10. shows the XSLT transformation design.

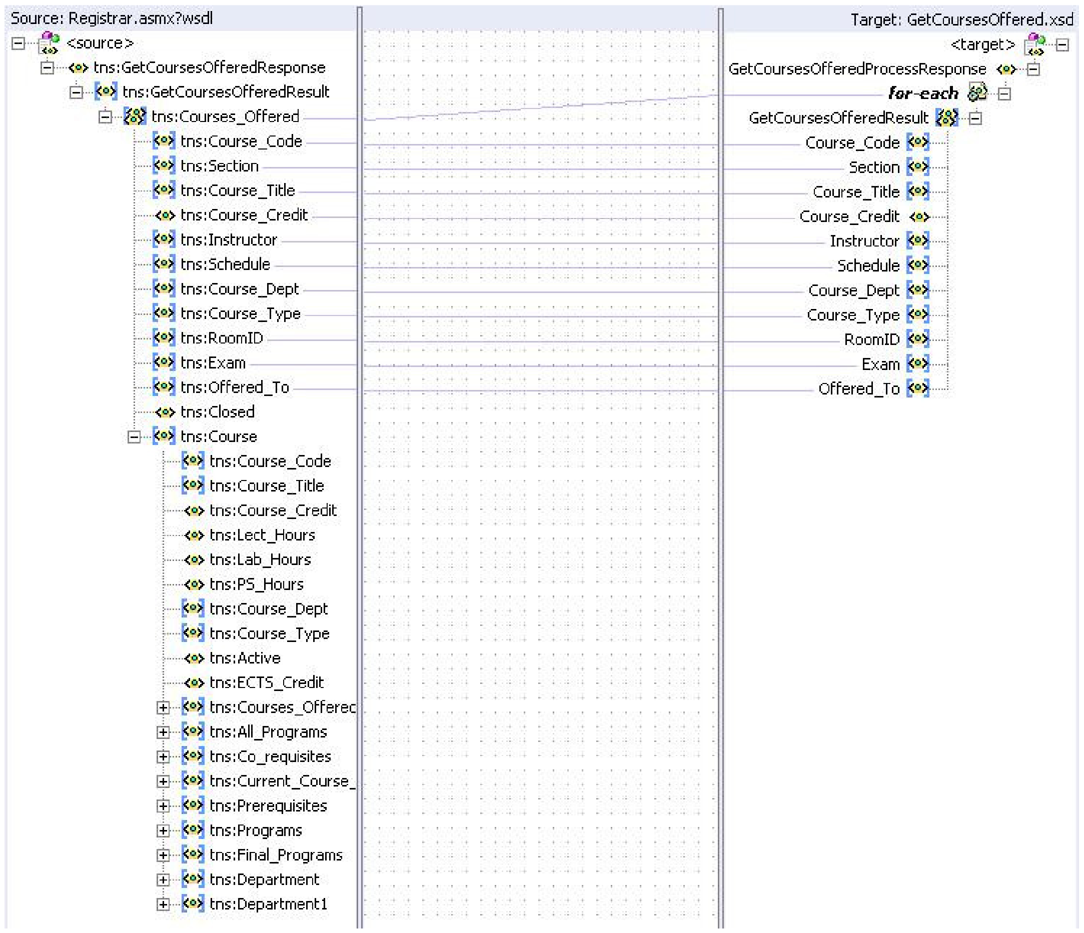


Figure 7.11. GetCoursesOffered XSLT Design

7.1.32 AddBook Service

This librarian service gets the details of the given book and adds it to the library catalogue. Service uses XSLT as the data transformation technique.

7.1.33 AddCourse Service

Registrar adds a course by giving the details with this service. Both XSLT and Oracle SOA Suite assignment system are used.

7.1.34 AddDepartment Service

Registrar adds new department to campus with this service. The input is transformed with the

XSLT and output is assigned with Oracle SOA Suite assignment.

7.1.35 AddFaculty Service

Registrar adds new faculty to campus with this service. The input is transformed with the XSLT and output is assigned with Oracle SOA Suite assignment.

7.1.36 AddStudent Service

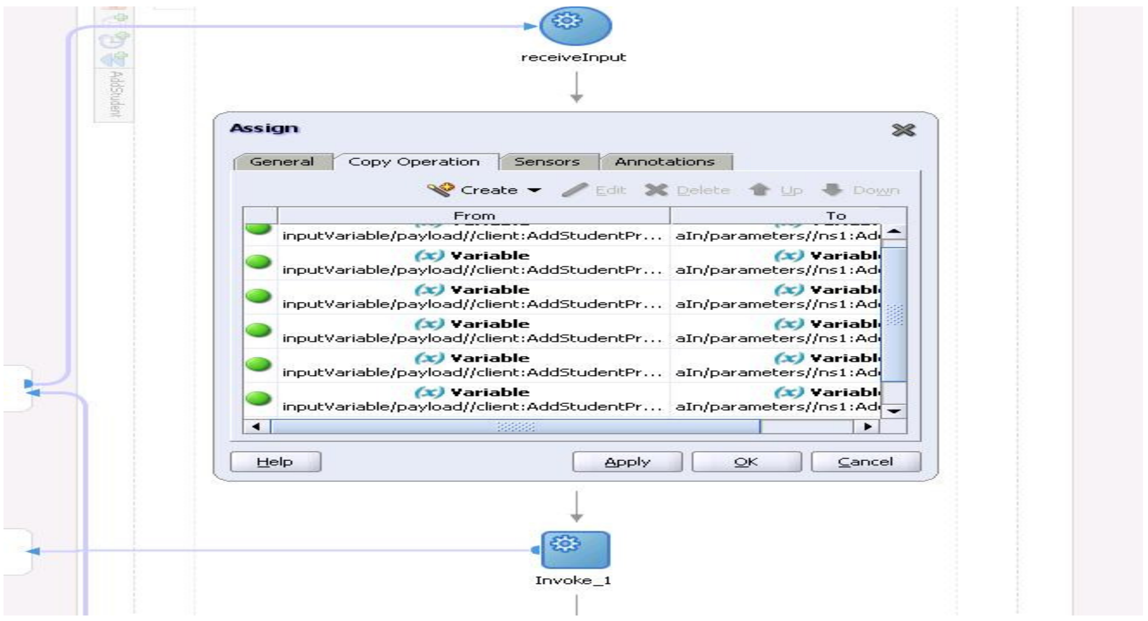
A new student is added to database with this registrar service. The input is transformed with the Oracle SOA Suite assignment system. Figure 7.11 shows this assignment.

7.1.37 AddStudentCourse

This service registers the student to the given course. This is not the final registration but the selection process. Figure 7.12 shows the transformation of input.

7.1.38 AddStudentPenalty Service

This service adds a penalty to the student. Penalties were defined in database before, so this service only binds it with the given student. Both assignment and XSLT were used.



	From	To
●	inputVariable/payload//client:AddStudentPr...	aIn/parameters//ns1:Ad
●	(x) Variable inputVariable/payload//client:AddStudentPr...	(x) Variabl aIn/parameters//ns1:Ad
●	(x) Variable inputVariable/payload//client:AddStudentPr...	(x) Variabl aIn/parameters//ns1:Ad
●	(x) Variable inputVariable/payload//client:AddStudentPr...	(x) Variabl aIn/parameters//ns1:Ad
●	(x) Variable inputVariable/payload//client:AddStudentPr...	(x) Variabl aIn/parameters//ns1:Ad
●	(x) Variable inputVariable/payload//client:AddStudentPr...	(x) Variabl aIn/parameters//ns1:Ad

Figure 7.12. AddStudent input transformation assignment

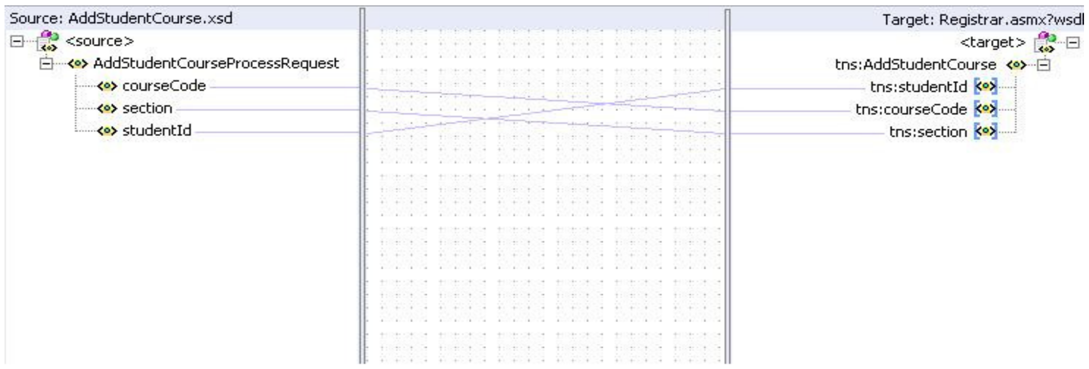


Figure 7.13. AddStudentCourse XSLT Design

7.1.39 ApproveStudentProgram service

This service is used by advisor to approve the proposed program by the student. If it can't add, it returns false.

7.1.40 CreateLogin Service

Site administrators use this service to create new logins to the campus system.

7.1.41 DeleteBook Service

Librarian deletes the given book from the catalogue with this service.

7.1.42 DoesStudentHaveDept Service

This financial department service is to check if the given student has paid it tuition for the current semester. Returns true if student hast paid anything.

7.1.43 DoesStudentHavePenalty Service

This service checks if the given student has any penalty from the faculty. If student has any penalty it return true.

7.1.44 DoesStudentHaveUnreturnedBooks Service

This service returns the count of the unreturned books that the student borrowed from the library.

7.1.45 DropSelectedCourse Service

If the student wants to drop a selected or registered course, he/she calls this service. The service deletes the course record from the student selected program.

7.1.46 GetCourseStudents Service

This service brings all students that take the specified course. Figure 7.13 show the XSLT design

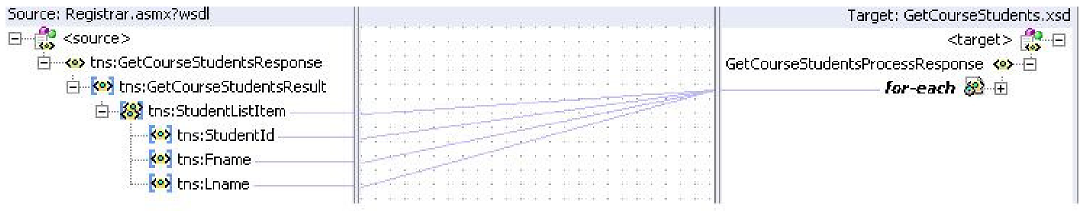

Source: Registrar.asmx?wsdl
<source>
tns:GetCourseStudentsResponse
tns:GetCourseStudentsResult
tns:StudentListItem
tns:StudentId
tns:Fname
tns:Lname
Target: GetCourseStudents.xsd
<target>
GetCourseStudentsProcessResponse
for-each

Figure 7.14. GetCourseStudent XSLT Design.

7.1.47 GetInstructorCourses

This service gets the list of the courses that the instructor conducts for the current semester. The list of the instructors is transformed using XSLT.

7.1.48 GetLogin Service

This service gives the detailed information about the login account. Assignment is used which can be seen in Figure 7.14.

7.1.49 GetPenalties Service

This service gets the list of the penalty types from the registrar web service. The call of the web service is shown in Figure 7.15.

7.1.50 GetStudentFinalCourses Service

Retrieves the final registered courses from the student program.

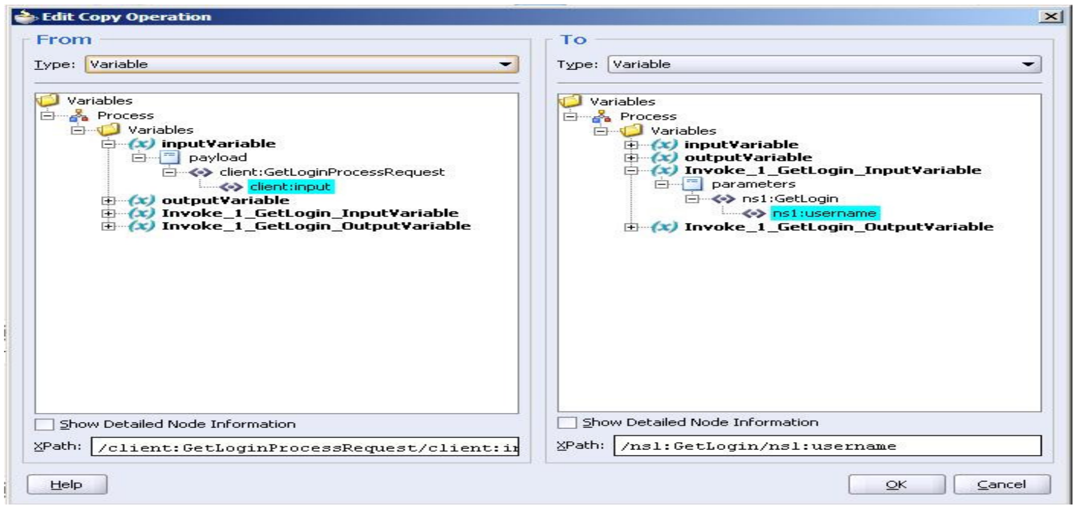

Edit Copy Operation
From
Type: Variable
Variables
Process
Variables
inputVariable
payload
client:GetLoginProcessRequest
client:input
outputVariable
Invoke_1_GetLogin_InputVariable
Invoke_1_GetLogin_OutputVariable
Show Detailed Node Information
XPath: /client:GetLoginProcessRequest/client:in
To
Type: Variable
Variables
Process
Variables
inputVariable
outputVariable
Invoke_1_GetLogin_InputVariable
parameters
ns1:GetLogin
ns1:username
Invoke_1_GetLogin_OutputVariable
Show Detailed Node Information
XPath: /ns1:GetLogin/ns1:username
Help
OK
Cancel

Figure 7.15. Design of assignment in Oracle SOA Suite

7.1.51 GetStudentFinalGrades Service

After the instructor starts to grade, it can choose from this service. ,

7.1.52 SubmitCourses Service

This service submits all the courses that student selected previously.

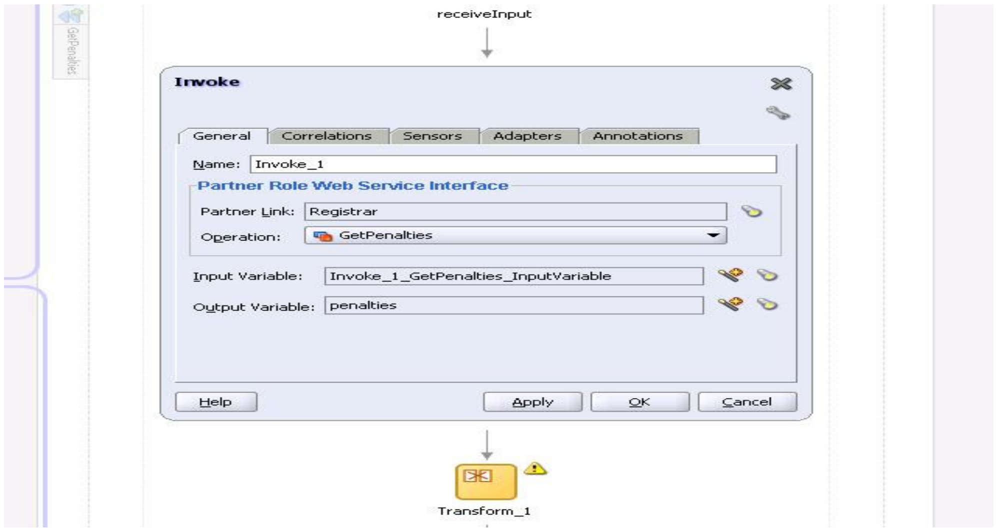


Figure 7.16. Webservice call setup window for GetPenalties

8. CONCLUSION

SOA, Service Oriented Architecture, is the architecture of a system to allow communicating and compliance of different platforms under a single environment that is open to highly reuse. In general, the services are brought together under the middle-tiers called ESP, Enterprise Service Bus, they work collaboratively without being aware of themselves with the help of SOA abstraction of the services. In the ESP, many routine jobs are contucted like logging, data transformation, orchestrating. SOA is not a product or a tool, it a technology independent architecture, an approach. It aims to orchestrate the functions to the services so that they scale well and can be reused. This way the component of the system is more independent and coarse-grained.

A product that is designed with SOA is expected to contain the following properties:

- Reusability
- Atomic structure
- Modularity
- Component based structure
- Interoperability
- Compliance with standards

Additionanlly, the a service that is provided through SOA should be compatible with the following properties:

- Capsullized
- Coarse-grained
- Self-descriptive
- Intelligent
- Quality
- Discoverable

SOA owns the interoperability to the web service technology. Web services bring the system functions online by using the SOAP, Simple Object Access Protocol. The intermessaging of the services is done with XML techonolgy. Web services are self-descriptive thorugh the WSDL, Web Service Definition Language, which is in XML format. A web service can be implemented on any platform and can be consumed from any other platform.

To better understand SOA and to see it in action, I have implemented a SOA based campus management system in which the actors open their services for others to consume. I tried to show that a university campus management system should be designed with SOA, because a university is a dynamic place where many parts co-operate frequenntly. In such a place, a management system without SOA will fail.

The majority of the time consumed on the project went to design of the Online Campus. SOA projects should take a considerable time to design because if the design of the services is not well, the system will get complex as it grows. I defined the roles, their tasks and what services they might release and consume. In the begginnig, I started to implement the system in Java. Then I changed the implementation platform to .NET. The reason for this change was to show the interoperability of different platforms under SOA. The Oracle SOA Suite is implemented under J2EE platform. So, the Online Campus system provides the interoperability between Java and .NET platforms. The database was chosen to be Microsoft SQL Server, to better operate with .NET platform. I used SQL to Linq (Language Integrated Query) techonology to work with the database. SQL to Linq creates classes of all the tables in the database and handles the query system. I queried the database through Linq syntax so no single classic SQL query was written.

I tried to keep the modularity high during the project. When a new department is added to the campus, it will be very easy to host since all the required information is available through the Oracle SOA Suite. The only concern would be integration.

Since a campus environment contains many departments that work simultaneously, when the workload is high, the system will need better hardware. For example if it is the registration date, registrar web service will be very busy. When you put better hardware, it might not help if it is a closed system. But in my system, it will be enough to put better hardware only to the busy web services.

Also security is very important in the projects like campus management systems because these systems are always face-to face with attackers. For example we can always find students who want to change their grades through the system security holes. This usually happens through SQL Injection attacks. However in my campus management system, the interface can not directly access the database. The interface should call only Oracle SOA Suite web services.

One thing to note that, Oracle SOA Suite is still very slow for concurrent connections. This drawback can be passed with hardware upgrades. But the designer interface of the services is not easy-to-use. Especillay, I had hard times while defining the web services and assigning the input variables. However, XSLT designer is very user-firendly and intelligent. For example, there is an auto mapping feture, that matches the resembling XML nodes automatically, which made my work easier when assigning complex web service variable. Another disadvantage of Oracle SOA Suite is that, it is hard to debug the deployed services. Sometimes it is hard to see where the service breaks and that caused the most painful times during the development.

In the future the project, I am planning to open the Online Campus to other services outside the campus. For example, the banking system can be included for the financial department of the campus so the financial deparment team can track all the transactions of students through the Online Campus interface. This requires the bank to provide web services for financial tasks. Another example is to provide interface for the instructors to the scientific databases. The instructors can track their papers thorugh Online Campus interface. This also requires the scientific database to open web services.

REFERENCES

Munindar P. Singh and Michael N. Huhns, (2005), "Service-Oriented Computing: Semantics, Processes, Agents", John Wiley & Sons, Ltd, West Sussex, England Oriented Computing.

Axel Buecker, Paul Ashley, Martin Borrett, Ming Lu, Sridhar Muppidi, (2007), "Understanding SOA Security Design and Implementation", Vervante.

Eric Newcomer, Greg Lomow, (2004), "Understanding SOA with Web Services", Addison Wesley.

Parichat Pasatcha, Komrhon Sunat, (2008), "A Distributed e-Education System Based on the Service Oriented Architecture".

Harry M. Sneed, (2006), "Integrating legacy Software into a Service oriented Architecture".

Ketil Lund, Anders Eggen, Dinko Hadzic, Trude Hafsøe, and Frank T. Johnsen, (2007), "Using Web Services to Realize Service Oriented Architecture in Military Communication Networks".

Parichat Pasatcha, Komrhon Sunat, (2008), "The Educational System Support Services Based on the Service Oriented Architecture".

Ho-Jun Lee, Jae-Woo Lee, Jeong-Oog Lee, (2008), "Development of Web services-based Multidisciplinary Design Optimization framework".

Francisco Garcı´a-Sa´nchez, Rafael Valencia-Garcı´a , Rodrigo Martı´nez-Be´jar, Jesualdo T. Ferna´ndez-Breis, (2008), "An ontology, intelligent agent-based framework for the provision of semantic web services ".

Anatoly Gladun, Julia Rogushina, Francisco Garcı´a-Sanchez,

Rodrigo Martı´nez-Be´jar, Jesualdo Toma´s Ferna´ndez-Breis, (2008), "An application of intelligent techniques and semantic web technologies in e-learning environments".

Steven Lynden, Arijit Mukherjee, Alastair C. Hume, Alvaro A.A. Fernandes, Norman W. Paton, Rizos Sakellariou, Paul Watson, (2008), "The design and implementation of OGSA-DQP: A service-based distributed query processor",

Pedro J. Muñoz-Merino, Carlos Delgado Kloos, Jesús Fernández Naranjo, (2008), "Enabling interoperability for LMS educational services".

Michiel Koning, Chang-ai Sun, Marco Sinnema, Paris Avgeriou, (2008), "VxBPEL: Supporting variability for Web services in BPEL".

Abdelghani Benharref, Rachida Dssouli, Mohamed Adel Serhani, Roch Glitho, (2008), "Efficient traces' collection mechanisms for passive testing of Web Services".

M. Gerz et al., (2006), "An Object-Oriented XML Schema for the MIP Joint Command, Control, and Consultation Information Exchange Data Model,".

I. Foster, C. Kesselman, S. Tuecke, (2001), “The anatomy of the grid: Enabling scalable virtual organizations”.

K. Gottschalk, S. Graham, H. Kreger, J. Snell, (2002), “Introduction to web services architecture”.

G. Aloisio, M. Cafaro, S. Fiore, (2005), “The grid-dbms: Towards dynamic data management in grid environments, in: Proc. Intl. Conf. on Information Technology: Coding and Computation”.

www.wikipedia.org (22 January 2009)

Westerkamp, P., (2006) , “Flexible Elearning Platforms: A Service-Oriented Approach”.

Nakwichian, Supotchanee and Sunetnanta, Thanwadee, (2003), "User-Centric Web Quality Assessment Model".

Omar, W. M. and Taleb-Bendiab, (2006), "Service oriented architecture for e-health support services based on grid computing".

Litoiu, M., (2004), “Migrating to Web Services a performance engineering approach Journal of Software Maintenance and Evolution”.

www.tibco.com, “Extending the Benefits of SOA beyond the Enterprise”

www.tibco.com, “Business Process Management on an SOA Foundation”

www.tibco.com, “ Architected Solution Delivery: Enhancing the Service Oriented Process”

www.tibco.com, “Data Integration Is Key to Realizing theFull Potential of SOA”

www.tibco.com, “Embracing SOA, The Benefits of Integration Independence”

www.tibco.com, “The Road to Enterprise-Class SOA”

www.tibco.com, “TIBCO SOA Governance Best Practises: An Introduction”

www.tibco.com, Lam, W. and Shankararaman, V. “Enterprise Architecture and Integration: Methods. Implementation and Technologies”

www.tibco.com, Lundberg, A. “Leverage Complex Event Processing to Improve Operational Performance”

www.tibco.com, “Leverage Complex Event Processing to Improve Operational Performance”

www.tibco.com, “SOA and Complexity”

www.oracle.com/technology/tech/soa/ uddi/index.html

www.oracle.com/technology/software/products/ jdev/index.html

www.oracle.com/technology/ products/ias/bpel/index.html www.oracle.com/technology/products/ integration/service-bus/index.html

www.oracle.com/technology/products/ias/ business_rules/index.html

www.oracle.com/technology/ products/webservices_manager/index.html

www.oracle.com/technology/tech/soa/ uddi/index.html

www.oracle.com/technology/products/ ias/index.html

Appendix I. IsStudentEligibleForRegistration BPEL

```
<process name="IsStudentEligibleForRegistration"
        targetNamespace="http://xmlns.oracle.com/IsStudentEligibleForRegistration"
        xmlns="http://schemas.xmlsoap.org/ws/2003/03/business-process/"
        xmlns:xp20="http://www.oracle.com/XSL/Transform/java/oracle.tip.pc.services.functions.Xpath20"
        xmlns:bpws="http://schemas.xmlsoap.org/ws/2003/03/business-process/"
        xmlns:ns4="fg"
        xmlns:ids="http://xmlns.oracle.com/bpel/services/IdentityService/xpath"
        xmlns:ldap="http://schemas.oracle.com/xpath/extension/ldap"
        xmlns:xsd="http://www.w3.org/2001/XMLSchema"
        xmlns:client="http://xmlns.oracle.com/IsStudentEligibleForRegistration"
        xmlns:ora="http://schemas.oracle.com/xpath/extension"
        xmlns:xref="http://www.oracle.com/XSL/Transform/java/oracle.tip.xref.xpath.XRefXPathFunctions"
        xmlns:hwf="http://xmlns.oracle.com/bpel/workflow/xpath"
        xmlns:ns1="http://xmlns.oracle.com/DoesStudentHaveDept"

xmlns:ehdr="http://www.oracle.com/XSL/Transform/java/oracle.tip.esb.server.headers.ESBHeaderFunctions"
        xmlns:ns3="http://xmlns.oracle.com/DoesStudentHaveUnreturnedBooks"
        xmlns:ns2="http://xmlns.oracle.com/DoesStudentHavePenalty"
        xmlns:bpelx="http://schemas.oracle.com/bpel/extension"
        xmlns:orcl="http://www.oracle.com/XSL/Transform/java/oracle.tip.pc.services.functions.ExtFunc">
  <!--
   //////////////////////////////////////////////////////////////////////////////////////////////////////////////
     PARTNERLINKS
     List of services participating in this BPEL process
   //////////////////////////////////////////////////////////////////////////////////////////////////////////////
  -->
  <partnerLinks>
    <!--
      The 'client' role represents the requester of this service. It is
      used for callback. The location and correlation information associated
      with the client role are automatically set using WS-Addressing.
    -->
    <partnerLink name="client"
             partnerLinkType="client:IsStudentEligibleForRegistration"
             myRole="IsStudentEligibleForRegistrationProvider"/>
    <partnerLink name="DoesStudentHaveDept"
             partnerRole="DoesStudentHaveDeptProvider"
             partnerLinkType="ns1:DoesStudentHaveDept"/>
    <partnerLink name="DoesStudentHavePenalty"
             partnerRole="DoesStudentHavePenaltyProvider"
             partnerLinkType="ns2:DoesStudentHavePenalty"/>
    <partnerLink name="DoesStudentHaveUnreturnedBooks"
             partnerRole="DoesStudentHaveUnreturnedBooksProvider"
             partnerLinkType="ns3:DoesStudentHaveUnreturnedBooks"/>
  </partnerLinks>
  <!--
   //////////////////////////////////////////////////////////////////////////////////////////////////////////////
     VARIABLES
     List of messages and XML documents used within this BPEL process
   //////////////////////////////////////////////////////////////////////////////////////////////////////////////
  -->
  <variables>
    <!-- Reference to the message passed as input during initiation -->
    <!-- Reference to the message that will be returned to the requester-->
    <variable name="inputVariable"
            messageType="client:IsStudentEligibleForRegistrationRequestMessage"/>
    <variable name="outputVariable"
```

```
          messageType="client:IsStudentEligibleForRegistrationResponseMessage"/>
  <variable name="penaltyIn"
          messageType="ns2:DoesStudentHavePenaltyRequestMessage"/>
  <variable name="penatltyOut"
          messageType="ns2:DoesStudentHavePenaltyResponseMessage"/>
  <variable name="replyFalse" type="xsd:boolean"/>
  <variable name="deptInput"
          messageType="ns1:DoesStudentHaveDeptRequestMessage"/>
  <variable name="deptOutput"
          messageType="ns1:DoesStudentHaveDeptResponseMessage"/>
  <variable name="bookIn"
          messageType="ns3:DoesStudentHaveUnreturnedBooksRequestMessage"/>
  <variable name="bookOut"
          messageType="ns3:DoesStudentHaveUnreturnedBooksResponseMessage"/>
  <variable name="check" type="xsd:boolean"/>
</variables>
<!--
  ////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////
   ORCHESTRATION LOGIC
   Set of activities coordinating the flow of messages across the
   services integrated within this business process
  ////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////
-->
<sequence name="main">
  <!-- Receive input from requestor. (Note: This maps to operation defined in
IsStudentEligibleForRegistration.wsdl) -->
  <receive name="receiveInput" partnerLink="client"
          portType="client:IsStudentEligibleForRegistration"
          operation="process" variable="inputVariable" createInstance="yes"/>
  <!-- Generate reply to synchronous request -->
  <assign name="assignInput">
    <copy>
      <from variable="inputVariable" part="payload"
          query="/client:IsStudentEligibleForRegistrationProcessRequest/client:input"/>
      <to variable="penaltyIn" part="payload"
         query="/ns2:DoesStudentHavePenaltyProcessRequest/ns2:input"/>
    </copy>
    <copy>
      <from expression="false()"/>
      <to variable="outputVariable" part="payload"
         query="/client:IsStudentEligibleForRegistrationProcessResponse/client:result"/>
    </copy>
    <copy>
      <from variable="inputVariable" part="payload"
          query="/client:IsStudentEligibleForRegistrationProcessRequest/client:input"/>
      <to variable="deptInput" part="payload"
         query="/ns1:DoesStudentHaveDeptProcessRequest/ns1:input"/>
    </copy>
    <copy>
      <from variable="inputVariable" part="payload"
          query="/client:IsStudentEligibleForRegistrationProcessRequest/client:input"/>
      <to variable="bookIn" part="payload"
         query="/ns3:DoesStudentHaveUnreturnedBooksProcessRequest/ns3:input"/>
    </copy>
  </assign>
  <invoke name="Invoke_1" partnerLink="DoesStudentHavePenalty"
         portType="ns2:DoesStudentHavePenalty" operation="process"
         inputVariable="penaltyIn" outputVariable="penatltyOut"/>
  <switch name="Switch_1">
```

```
   <case
condition="boolean(bpws:getVariableData('penatltyOut','payload','/ns2:DoesStudentHavePenaltyProcessRespon
se/ns2:result'))">
    <sequence name="Sequence_1">
     <assign name="Assign_8">
      <copy>
       <from variable="inputVariable" part="payload"
          query="/client:IsStudentEligibleForRegistrationProcessRequest/client:input"/>
       <to variable="deptInput" part="payload"
         query="/ns1:DoesStudentHaveDeptProcessRequest/ns1:input"/>
      </copy>
     </assign>
    </sequence>
   </case>
   <otherwise>
    <sequence name="Sequence_2">
     <assign name="Assign_9">
      <copy>
       <from
expression="boolean(bpws:getVariableData('penatltyOut','payload','/ns2:DoesStudentHavePenaltyProcessRespo
nse/ns2:result'))"/>
       <to variable="check"/>
      </copy>
     </assign>
     <reply name="Reply_1" partnerLink="client"
        portType="client:IsStudentEligibleForRegistration"
        operation="process" variable="outputVariable"/>
     <terminate name="Terminate_1"/>
    </sequence>
   </otherwise>
  </switch>
  <invoke name="Invoke_2" partnerLink="DoesStudentHaveDept"
      portType="ns1:DoesStudentHaveDept" operation="process"
      inputVariable="deptInput" outputVariable="deptOutput"/>
  <switch name="Switch_2">
   <case
condition="boolean(bpws:getVariableData('deptOutput','payload','/ns1:DoesStudentHaveDeptProcessResponse/
ns1:result'))=false()">
    <sequence name="Sequence_3">
     <reply name="Reply_2" partnerLink="client"
        portType="client:IsStudentEligibleForRegistration"
        operation="process" variable="outputVariable"/>
     <terminate name="Terminate_2"/>
    </sequence>
   </case>
   <otherwise>
    <assign name="Assign_4">
     <copy>
      <from variable="inputVariable" part="payload"
         query="/client:IsStudentEligibleForRegistrationProcessRequest/client:input"/>
      <to variable="bookIn" part="payload"
        query="/ns3:DoesStudentHaveUnreturnedBooksProcessRequest/ns3:input"/>
     </copy>
    </assign>
   </otherwise>
  </switch>
  <invoke name="Invoke_3" partnerLink="DoesStudentHaveUnreturnedBooks"
      portType="ns3:DoesStudentHaveUnreturnedBooks" operation="process"
      inputVariable="bookIn" outputVariable="bookOut"/>
  <assign name="Assign_5">
    <copy>
      <from variable="bookOut" part="payload"
            query="/ns3:DoesStudentHaveUnreturnedBooksProcessResponse/ns3:result"/>
      <to variable="outputVariable" part="payload"
          query="/client:IsStudentEligibleForRegistrationProcessResponse/client:result"/>
    </copy>
  </assign>
  <switch name="Switch_3">
    <case
condition="boolean(bpws:getVariableData('outputVariable','payload','/client:IsStudentEligibleForRegistrationPr
ocessResponse/client:result'))=false()">
      <assign name="Assign_7">
        <copy>
          <from expression="false() "/>
          <to variable="outputVariable" part="payload"
              query="/client:IsStudentEligibleForRegistrationProcessResponse/client:result"/>
        </copy>
      </assign>
    </case>
    <otherwise>
      <sequence name="Sequence_4">
        <assign name="Assign_6">
          <copy>
            <from expression="true()"/>
            <to variable="outputVariable" part="payload"
                query="/client:IsStudentEligibleForRegistrationProcessResponse/client:result"/>
          </copy>
        </assign>
      </sequence>
    </otherwise>
  </switch>
  <reply name="replyOutput" partnerLink="client"
         portType="client:IsStudentEligibleForRegistration"
         operation="process" variable="outputVariable"/>
 </sequence>
</process>
```

Appendix II. GetCoursesOfferedResponse XSL Ttransformation

```
<xsl:stylesheet version="1.0" xmlns:mime="http://schemas.xmlsoap.org/wsdl/mime/"
            xmlns:bpws="http://schemas.xmlsoap.org/ws/2003/03/business-process/"
            xmlns:wsdl="http://schemas.xmlsoap.org/wsdl/"
            xmlns:soap12="http://schemas.xmlsoap.org/wsdl/soap12/"

xmlns:ehdr="http://www.oracle.com/XSL/Transform/java/oracle.tip.esb.server.headers.ESBHeaderFunctions"
            xmlns:tns="http://tempuri.org/"
            xmlns:http="http://schemas.xmlsoap.org/wsdl/http/"
            xmlns:ns0="http://www.w3.org/2001/XMLSchema"
            xmlns:hwf="http://xmlns.oracle.com/bpel/workflow/xpath"
            xmlns:soap="http://schemas.xmlsoap.org/wsdl/soap/"
            xmlns:s1="http://microsoft.com/wsdl/types/"
            xmlns:xp20="http://www.oracle.com/XSL/Transform/java/oracle.tip.pc.services.functions.Xpath20"
            xmlns:xref="http://www.oracle.com/XSL/Transform/java/oracle.tip.xref.xpath.XRefXPathFunctions"
            xmlns:soapenc="http://schemas.xmlsoap.org/soap/encoding/"
            xmlns:xsl="http://www.w3.org/1999/XSL/Transform"
            xmlns:ora="http://schemas.oracle.com/xpath/extension"
            xmlns:ids="http://xmlns.oracle.com/bpel/services/IdentityService/xpath"
            xmlns:orcl="http://www.oracle.com/XSL/Transform/java/oracle.tip.pc.services.functions.ExtFunc"
            xmlns:ns1="http://xmlns.oracle.com/GetCoursesOffered"
            xmlns:tm="http://microsoft.com/wsdl/mime/textMatching/"
            exclude-result-prefixes="xsl mime wsdl soap12 tns http ns0 soap s1 soapenc tm ns1 bpws ehdr hwf
xp20 xref ora ids orcl">
  <xsl:template match="/">
    <ns1:GetCoursesOfferedProcessResponse>
      <xsl:for-each select="/tns:GetCoursesOfferedResponse/tns:GetCoursesOfferedResult/tns:Courses_Offered">
        <ns1:GetCoursesOfferedResult>
          <ns1:Course_Code>
            <xsl:value-of select="tns:Course_Code"/>
          </ns1:Course_Code>
          <ns1:Section>
            <xsl:value-of select="tns:Section"/>
          </ns1:Section>
          <ns1:Course_Title>
            <xsl:value-of select="tns:Course_Title"/>
          </ns1:Course_Title>
          <ns1:Course_Credit>
            <xsl:value-of select="tns:Course_Credit"/>
          </ns1:Course_Credit>
          <ns1:Instructor>
            <xsl:value-of select="tns:Instructor"/>
          </ns1:Instructor>
          <ns1:Schedule>
            <xsl:value-of select="tns:Schedule"/>
          </ns1:Schedule>
          <ns1:Course_Dept>
            <xsl:value-of select="tns:Course_Dept"/>
          </ns1:Course_Dept>
          <ns1:Course_Type>
            <xsl:value-of select="tns:Course_Type"/>
          </ns1:Course_Type>
          <ns1:RoomID>
            <xsl:value-of select="tns:RoomID"/>
          </ns1:RoomID>
          <ns1:Exam>
            <xsl:value-of select="tns:Exam"/>
          </ns1:Exam>
          <ns1:Offered_To>
            <xsl:value-of select="tns:Offered_To"/>
          </ns1:Offered_To>
        </ns1:GetCoursesOfferedResult>
      </xsl:for-each>
    </ns1:GetCoursesOfferedProcessResponse>
  </xsl:template>
</xsl:stylesheet>
```

Appendix III. An example Oralce SOA generated WSDL

```
<definitions name="GetStudentAddressInformation"
targetNamespace="http://xmlns.oracle.com/GetStudentAddressInformation"
xmlns="http://schemas.xmlsoap.org/wsdl/"
xmlns:client="http://xmlns.oracle.com/GetStudentAddressInformation"
xmlns:plnk="http://schemas.xmlsoap.org/ws/2003/05/partner-link/">

<!-- ~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
TYPE DEFINITION - List of services participating in this BPEL process
The default output of the BPEL designer uses strings as input and
output to the BPEL Process. But you can define or import any XML
Schema type and use them as part of the message types.
~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~ -->
<types>
<schema xmlns="http://www.w3.org/2001/XMLSchema">
<import namespace="http://xmlns.oracle.com/GetStudentAddressInformation"
schemaLocation="GetStudentAddressInformation.xsd" />
</schema>
</types>

<!-- ~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
MESSAGE TYPE DEFINITION - Definition of the message types used as
part of the port type defintions
~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~ -->
<message name="GetStudentAddressInformationRequestMessage">
<part name="payload" element="client:GetStudentAddressInformationProcessRequest"/>
</message>
<message name="GetStudentAddressInformationResponseMessage">
<part name="payload" element="client:GetStudentAddressInformationProcessResponse"/>
</message>
<!-- ~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
PORT TYPE DEFINITION - A port type groups a set of operations into
a logical service unit.
~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~ -->
<!-- portType implemented by the GetStudentAddressInformation BPEL process -->
<portType name="GetStudentAddressInformation">
<operation name="process">
<input  message="client:GetStudentAddressInformationRequestMessage" />
<output message="client:GetStudentAddressInformationResponseMessage"/>
</operation>
</portType>
<!-- ~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
PARTNER LINK TYPE DEFINITION
~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~ -->
<plnk:partnerLinkType name="GetStudentAddressInformation">
<plnk:role name="GetStudentAddressInformationProvider">
<plnk:portType name="client:GetStudentAddressInformation"/>
</plnk:role>
</plnk:partnerLinkType>
</definitions>
```

CURRICULUM VITAE

Murat Kaya

22/06/1978 FRANSA

E-mail: mkaya78@gmail.com

GSM: 0 535 817 44 87

Address: Eminalipaşa cad. Kuru Dere Sok. Çam Apt. No: 28/4 Suadiye / İstanbul

Education

M.Sc. in Computer and Information Technology Sciences, not graduated, Doguş University (%100 Scholarship), Institute of Science and Technology, Istanbul – Turkey. (2006 - still attending with gpa: 3,88)

B.Sc. in Computer Engineering, 2002 - 2005, Işık University (ÖSYM Scholarship) , Faculty of Engineering, Istanbul – Turkey. (GPA:3.47, Rank:4)

1992 - 1996, Kadir Has Anatolian High School, Istanbul – Turkey.