SERVICE-ORIENTED INTEGRATION OF INFORMATION SYSTEMS FOR LOGISTICS MANAGEMENT

A Thesis Submitted to
the Graduate School of Engineering and Sciences of
İzmir Institute of Technology
in Partial Fulfillment of the Requirements for the Degree of

MASTER OF SCIENCE

in Computer Software

by
Şevket ÇETİN

December 2013

İZMİR

We approve the thesis of **Şevket Çetin**

Examining Committee Members:

Asst. Prof. Dr. Tolga AYAV
Department of Computer Engineering, Izmir Institute of Technology

---

Asst. Prof. Dr. Tuğkan TUĞLULAR
Department of Computer Engineering, Izmir Institute of Technology

---

Asst. Prof. Dr. Derya BİRANT
Department of Computer Engineering, Dokuz Eylül University

5 December 2013

Asst. Prof. Dr. Tolga AYAV
Supervisor, Department of Computer Engineering
Izmir Institute of Technology

Prof. Dr. İ. Sıtkı AYTAÇ
Head of Department of Computer
Engineering

Prof. Dr. R. Tuğrul SENGER
Dean of Graduate School of
Engineering and Sciences

ACKNOWLEDGEMENT

Foremost, I would like to express my sincere gratitude to my advisor Asst. Prof. Dr. Tuğkan TUĞLULAR for the continuous support of my study and research, for his patience, motivation, enthusiasm, and immense knowledge. His leadership helped me in all study and writing my thesis.

I want to thank to manager of Bimar cooperation Murat ÖZEMRE for his support on thesis project.

I wish to express my sincere thanks to my friend and project partner Dilek TİTİZ for his support and friendship during thesis work.

Also I would like to thank to Asst. Prof. Dr. Tuğkan TUĞLULAR and Bimar Cooperation. With their help, this project case study work is submitted as a SANTEZ project with project number 00933-STZ.2011-1.

I also want to thank to developers of Bimar; Murat FERUZ, Engin GÜRAY, Kerim ÖNDER and Uğur DOĞAN for their helps about technical details during thesis study.

ABSTRACT

SERVICE-ORIENTED INTEGRATION OF INFORMATION SYSTEMS FOR LOGISTICS MANAGEMENT

Developments in information technology have become more crucial for corporate firms and businesses. They make use of this technology to manage business processes and it is one of the most invested domains by corporations. As technical infrastructures of companies improve, the number of enterprise-oriented and special software developed for business processes increase, too. With the augmentation of cooperation between companies and incorporated business processes, in time, a need for integration emerges for the applications running in diverse infrastructures and technologies.

Logistics business processes are a part of a business domain where there are multiple areas of study such as railway, seaway, road transportation and depot, and where multiple companies and a high number of customer needs are managed. Integrations between companies should be quick, reliable, easily-adaptable to changing business processes is a crucial requirement. As integrations play a significant role in the management of process, the correct establishment of the integration architecture, convenience for follow-ups and management are critical for the flow of business processes related to the monitoring of the possible problems.

The main point of this thesis is based on a need for a software infrastructure that will enable integrations to work together. Thus, by getting integrations to utilize service-based architecture, to react quickly to changing business processes and customer needs, it is aimed to provide management and for exception monitoring. That’s why I focused on integration of service-based information systems for logistics management in my thesis.

ÖZET

LOJİSTİK YÖNETİMİ İÇİN SERVİS TABANLI BİLGİ SİSTEMLERİ ENTEGRASYONU

Son yıllarda bilgi teknolojilerinin gelişmesi, kurumsal firmalar ve işletmelerin iş süreçlerinin yönetebilmek için kullandıkları ve yatırım yaptıkları alanlardan birisidir. Şirketlerin teknik altyapılarını geliştirmesiyle birlikte, iş süreçlerine özel yazılımlar, kurumlarınaa yönelik geliştirilen uygulamaların sayısı artmaktadır. Kurumlar ve işletmelerin birbirleriyle ilişkileri ve birleştirilen iş süreçleri nedeniyle farklı altyapı ve teknolojilerdeki bu uygulamaların entegrasyonu ihtiyacı ortaya çıkmaktadır.

Lojistik iş süreçleri de kara, deniz, hava, demir yolu, depo gibi fazla sayıda çalışma alanı içeren, farklı firma ve fazla sayıda müşteri taleplerinin yönetildiği bir iş alanıdır. İş süreçlerinin kompleks, birlikte çalışılan firma ve kurum sayısının çok olması sebebiyle entegrasyonu yapılacak uygulamarın sayısı da oldukça fazladır. Bu entegrasyonların hızlı, güvenilir, değişen iş süreçlerine çabuk adapte olması ise kritik bir gereksinimdir. Sürecin yönetilmesinde entegrasyonların rolü büyük olduğundan, entegrasyon mimarisinin doğru kurulması, izleme ve yönetimin rahat yapılabilmesi, oluşabilecek hataların takibi iş akışlarının düzenli olarak çalışması açısından kritiktir.

Tez kapsamını da lojistik iş süreçleri için entegrasyonların birlikte çalışmasını sağlayan bir yazılım altyapı gereksinimi tezin ana oluşturmuştur. Bu sayede entegrasyonları servis tabanlı mimari'nin avantajlarından faydalanmış, değişen iş süreçlerine ve müşteri taleplerine hızlı cevap verme, entegrasyonların yönetimi ve süreç içinde yaşanan hataların takibi için gerekli çözümü sağlayacaktır. Bu sebeple lojistik yönetimi için bilgi sistemlerinin servis tabanlı entegrasyonu bu tezin konusunu oluşturmaktadır.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

CHAPTER 1

INTRODUCTION

In today's world, it has been an ordinary fact that all enterprises around the world cooperate with each other. This has become a necessity for them to grow, to be successful in the market, thus to increase their profits.

With the increasing number of companies working together, that the processes run in cooperation plays a much more important role because with cooperation and partnership, a flexible action area and more advantages for the company are obtained. That's why management departments in companies aim to increase the number of providers and the enterprises that they will cooperate with in order to get a wider business network.

Integration between companies plays an important role for healthy processes carried out in cooperation and for the companies to communicate with each other. Considering this, to achieve cooperability of different software used in a single enterprise, dozens of integrations need to be defined and managed. This is only possible with an integration infrastructure that can provide an optimum solution and adapt to different technological infrastructures of the enterprises.

Amongst each other, integration architectures can work in different topologies according to their diverse working principles and architectures. In this study, the aim is to use message-based integration architecture which will provide the optimum solution.

The focal point of this study is a Service-based Integration Infrastructure for information systems of the logistics processes where there is multiple companies work together and business processes are complicated. This joint study by Izmir Institute of Technology, by Arkas which is a logistics company and by Bimar managing the IT processes of Arkas has been accepted as a Sanayi Tezleri Programı (SAN-TEZ) project. With this study, an integration infrastructure that will enable cooperation with logistics business processes aims a Business Activity Framework allowing the management of business data.

With this architecture, shared business processes that Arkas Company, which is in the container logistics sector, carries out with other enterprises and providers will be co-operable and monitorable. Thus, their cooperation level which now cannot be improved due to this technological infrastructural short-coming will increase and the sector will be more efficient and more dynamic.

Container Logistics Sector manages transportation activities in many mods (seaway, road, railway, air, depot, port, etc.). Since the number of shipping areas and options is high, the number of cooperating enterprises and providers is also supreme. A shipment or transportation action is realized as a business process to which more than one enterprise contributes. In order for business processes to run healthily, integration should be manageable and monitorable.

However, high number of integrations is one of the crucial parameters for the solution of the problem because the integrations examined in the study work as different integrations although they represent the same processes and use similar data. This situation increases the cost both during the development and the following maintenance and support processes of the integrations. Furthermore, increasing number of integrations affect customer satisfaction with regards to follow-ups and maintenance problems.

With the integration infrastructure developed in this study, the problems aforementioned and the exceptions ensuring these problems will be monitorable from a single point. It will easier to detect and to deal with the occurring problems and also more convenient to determine which problems are critical. Thus, this study aims to provide a solution to the relating problems with an architecture that will enable enterprises to communicate with each other during these processes. The main approach used to create this architecture was Enterprise Service Bus architecture (ESB) which supports Message Based Middleware and Service Oriented Architecture (SOA) features. SOA architecture is an approach where logic and infrastructure resources are organized as services for business processes and accessed through mutual message exchange. By utilizing SOA properties with this architecture, the aim is to provide integration of complex business processes with reusable services. The route to solution is through integration services developed as agile, flexible and quick to respond changing business flows and customer needs.

For the services targeted within the scope of the study, a Microsoft product, BizTalk integration server and ESB Toolkit which is released for ESB architecture support will be used. Although BizTalk integration server used in Bimar Company is message-based as the working principle, it cannot make use of SOA architecture properties. The increasing number of integrations, changing business processes and integrations needed to be developed enhances the complexity of the present system. This increasing complexity costs more time and money to companies and complicates the solutions to the occurring problems.

In the thesis, the aim is to provide a service-based integration of logistics information systems through analyses of container logistics processes, data modeling and through a software infrastructure which the integration can cooperate with. The advantages provided with this study are the following.

- Realization of integrations in a service-oriented architecture.
- Reduction in the number of present integrations and providing a reusable infrastructure for new integrations.
- Reduction in time spent on development and maintenance
- Providing a service infrastructure for the follow-up of integrations and the problems arisen in the business processes.
- Providing an integration which is more agile for changing business processes and customer needs.

CHAPTER 2

REVIEW OF LITERATURE

The rise of cooperation between companies and enterprises that develop information technologies infrastructure has brought about a need for communication for the applications that work in different infrastructures. With the increasing number of diverse applications, a lot of efforts have been put into to improve communication and cooperation atmosphere, to reduce management costs and to optimize all processes. Research on these topics is outlined below.

The study which was explained in the article *An Integration Research on Service-oriented Architecture (SQA) for Logistics Information System* by Luyang Zhang, Jiaqi Li, Ming Yu focuses on the management of container logistics business processes with ESB that supports a SOA-based architecture. Here they determined that this was the most effective methodology for the integration solutions of service-based architectures. Container logistics business processes have a huge area of business and within themselves they contain diverse enterprise applications and business processes. That's why, he mentions about an emerging need for the applications from different platforms to work in an integrated way. In this integration architecture, this methodology was used for ESB application: each running present application is integrated to the system as one web-service, as applications added as we-services can operate on their own, they obtain a reusable property and can adapt more to changing business processes. Not a single application will directly communicate with another, but the integrations will be realized on ESB backbone. In this solution offer, up to 60% reduction in development costs is reported thanks to ESB architecture and service usage [25].

In their article *Research and Application of the ESB Based on Agent in the Integration of the MIS in Power Plant*, Fei and Shufen propose to use ESB and SOA architecture in management information systems of power plants for example. In these management systems, system integration is targeted by forming a ESB backbone for the domain which consist of a financial system, a human resources management system, a production management system, a scheduling management system, material and equipment management systems [26].

Furthermore in this system integration, first of all, present systems are determined as services and are included in the system. Through a different approach, systems that should be integrated with each other can work under sub-ESBs by utilizing more ESB structure. In this way, applications running only under its sub-ESB architecture check the incoming requests in sub-ESB for the first time and sends them to request agent ESB to find the responsive service if the response does not returns. An ESB structure that supports SOA architecture can also be used for different purposes e.g. for common platforms in schools as explained in the article *Research on application of Web based ESB in School Common Data Platform*. The usage of web services are realized within ESB, data interchange between different departments are provided on Web Services Description Language (WSDL). Heterogeneous systems that are outside the school applications are adapted to ESB with adapters which need to be developed. As various protocols are supported, adapters to be implemented can also be used in different protocols. In this study, integration or data resemblance is not checked, yet each of end-point is included in the system as an integration point [27].

One of the methods for the management of business process solutions and management of integrations is explained in this study by Rajini and Bhuvaneswair. As explained in *Service Based Architecture for Manufacturing Sector,* architecture was designed consisting of five layers and each layer manages a part of the process: Presentation Layer where other layers put user products and services into the system, Business Process Layer which manages business processes where all enterprises or applications connect to the system, Business Service Layer which connects business processes with integration layer, Integration Layer which handles service integration, routing and transformation processes; and Data Layer where data used by the services are stored and where physical resources that can be accessed on the Internet are located. With this architecture, the aim was to resolve process integrations with service-based architecture, However, there is not a systems present to deal with exception or integration monitoring [28].

Operating Enterprise Application Integration with Enterprise Service Bus was studied in *Research of Enterprise Application Integration (EAI) Based-on ESB* by Tao, and Wu. The article explains that applications under EAI-ESB approach communicate with each other through an architecture consisting of eight steps. These steps are:

Message construction, Messaging channels, Listener, Decryptor, Validator, Enricher, Transformer and Router. Moreover, here it is explained that exceptions have two different exception handling strategies: First strategy is the exception handling occurring in the connection points of the end-points. Second one is the exception handling on the basis of applications which components, namely, business processes work with. Exception framework and service which I applied in my thesis, enables the use of exception handling infrastructure for both strategies [29].

Container logistics business processes can be analyzed in a way to be able to work in SOA architecture, as it was done in the study by Wolfgang Seiringer. In his article *Service-oriented Analysis of Logistics Services,* and the attempts to explain with which methods business processes are defined as services. There two service definitions according to study, first of which is Web-Service standard WDSL and second one define service from three points of view: Service value independent of technology, service offering and service process. In the service analysis of container logistics domain, the methodology how to determine the similarities between the entities that are defined as service value was explained. This methodology involves two steps: first being the determining and analysis of the present services and the second one being entity similarity study and service modeling [30].

The concept of agility has also been influential in determining dynamic routing processes. Yo and Yan, in their article, *Towards the Integration of Enterprise Service Bus with Universal Description Discovery and Integration (UDDI) Server: A Case Study* suggest that the agility concept which ESB benefits from the advantages of SOA architecture can be developed to enable dynamic and in run-time routing for the routing process within the architecture. In this methodology, business processes are managed in Business Layer, and there is a UDDI server in the integration layer. ESB uses UDDI server for message routing. In this way, with proxy server, ESB sends a request to UDDI server for message routing it receives and incoming request includes the address to which the message leads. Thus, this enables an agile infrastructure in rum-time, providing dynamic routing. In our study, UDDI infrastructure is present in ESB. However, in the case study of Bimar, we used Business Rule Engine infrastructure for dynamic routing [31].

When the studies conducted are examined, it is seen that a solution is targeted for the integration of applications and the management process of these applications. For solving the diverse business flows and related problems, service-based architecture use is suggested. In other studies parallel with this thesis, integrations are aimed through reusable services, and Enterprise Service Bus architecture are used in different ways to provide solutions to the problems. In order to provide a quick solution to monitorability and possible problems, exception handling is engaged in ESB architecture. Like the studies carried out before, in this study, utilizing the SOA advantages, an effective solution is targeted on ESB architecture, furthermore, launching a product which will provide follow-up and control with regards to service and business flows is aimed.

CHAPTER 3

BACKGROUND

3.1. Enterprise Application Integration

Enterprise Application Integration (EAI) enables various software systems or applications to intercommunicate. Furthermore, subsystems are put together with the help of enterprise application integration, and they act as a single system. Thus they are able to function as a coordinated, whole body. These applications may be available out in the market for private usage, or they might as well be developed by the company itself.

The concept of integration of enterprise application is the direct result of the communication challenge that enterprises or companies needed to intercommunicate. The communication in question does have to be directed towards a business goal and it has to be attained regardless of temporal and special features of the applications as itwas nicely put by Samtani and Sadhwani as "*the process of creating an integrated infrastructure linking disparate systems, applications, and data sources within a corporate enterprise*" [1].

EAI is a six step process: The first step is the receiving the message, then this message is altered, and translated. The fourth step is the routing of the message. Directing of the message to the desired spot and business process management are the last two steps. The temporal process of the delivery of the messages is dependent on the businesses.

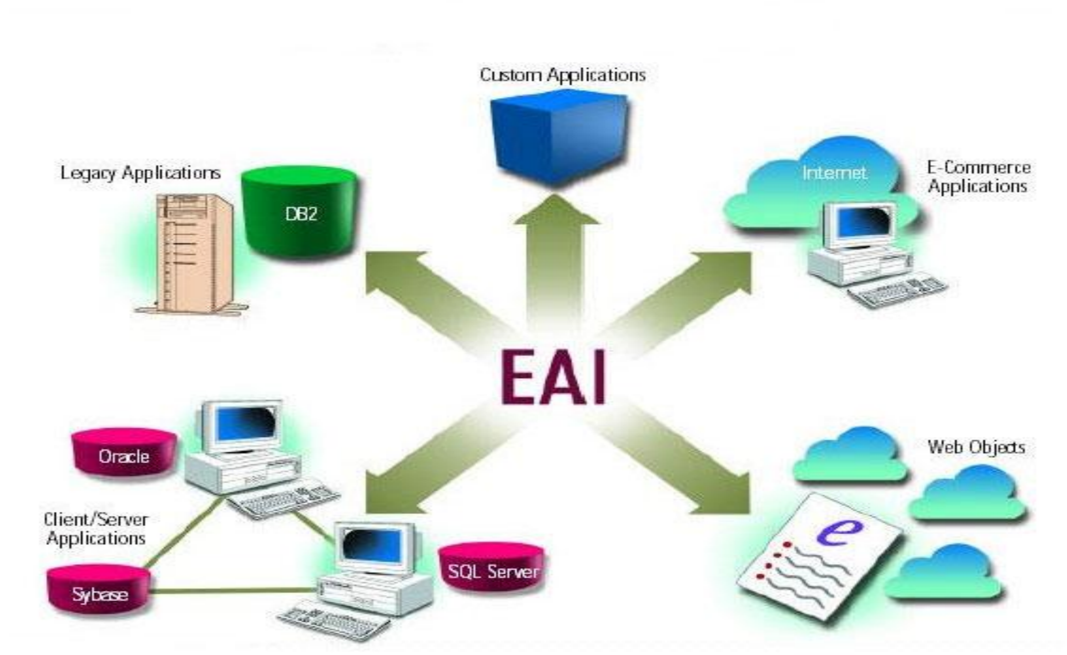


Figure 3.1. Enterprise Integration Systems

3.1.1. Reasons for Integration of Information Systems

With the advent of computationalization and with the structural challenges faced, organizations have depended on information systems, these organizations, in time, came up with particular individual systems, which meant that these systems were bound to be different from one another [2].

At first, ERP (enterprise resource planning) vendors succeeded in providing extensive business applications, yet with the rise of technology, the number of supplier/vendor companies and the number of companies that were involved in in-house development has risen, and naturally along with these, so has the number of the systems that have been developed by these companies, thus making these systems insufficient thereby creating a need for integration.

These systems in question include a number of applications which are customized according to the purpose they serve for and are designed to operate in various operating system platforms or media. Yet, these systems are formed in a way that they can only address to the particular tasks in a particular area. Hence in time focusing on their functionality has culminated in the creation of "islands of applications", disconnecting the systems from each other as Sawhney explains [3].

It is essential that these applications should be integrated with each other and this is only possible with supporting common business processes and data sharing amongst the applications. However, the integration has to sustain an effective, safe data exchange between applications used by the enterprises. With the integration of these systems, the integration that is possible between other service providers and the clients has the potential to bring about a wider area of actions and the advantage of competition for the companies [4]. Focal point in this thesis, container logistic information systems, is a perfect example for distributed applications and different platforms. Different business processes have to be in operation all together and supplier company systems, other in-house services or major billing systems like SAP need to be present in business processes. These two factors require integrity of reliable and stable integrations.

3.1.2. Enterprise Integration Challenges

It is a strenuous task to integrate enterprise applications and distributed systems because these systems are seen as a very valuable investment both financially and with regard to the amount of data they contain for the company, hence, making it almost impossible to alter them with newer systems [5].

Enterprise integration has to be constructed in a way that can handle numerous applications operating on various platforms in various places. Enterprise application integration suites are provided by software vendors and these suites enable integration between different languages and platforms, and connect to numerous business applications. Nevertheless, the technical base renders only a small amount of the challenges faced in the integration process which goes beyond corporate and technical issues.

Corporations are usually required to alter their policies when it comes to enterprise integration. Corporate applications are usually concentrated on a particular functional realm. Customer Relationship Management (CRM), Billing, Finance or Logistic Information Systems that is the focal point of my thesis are main examples. Effective enterprise integration is possible only with intercommunication between numerous computer systems and also with communication created between business units and IT departments. Applications are not controlled one by one by integration application groups since each and every application becomes a member of a body of an integrated applications and services.

An application might have a wide range of functionality i.e. a payment system application, thus it needs to provide an interface enabling communication between all vendors also to fulfill customer needs, yet integration brings about extensive ramifications on the businesses. When an integration solution is adopted into one of the most important business processes like payment system functions, solution affects progress of the whole business. A failure in an integration solution may lead to lost orders, misrouted payments as most importantly customer dissatisfaction, thus costing businesses millions of dollars.

When developing integration solutions, developers generally face numerous challenges one of which is their limited control on the applications that participate in the system. Some of those applications might be packaged or may just be used for a long time and hard to alter, which makes them very hard to adapt to an integration solution. Yet it is often easier to implement a part of the solution within the application nevertheless it is very probable that all integration objectives, especially ones that are very hard to adapt to applications, might not be reached. As the systems improve and new platforms, services and applications emerge in the market, the number of incompatible areas or points in the integration process will continue to increase, thus making it impossible to change all the failed points in the integration process and to make them compatible again.

Although there is a huge demand for integration solutions, only a handful of standards are seen as legitimate in the domain. XML, XSL and Web services can be demonstrated as the most advanced steps towards standardized features for integration solutions. However, with the advances in technology, changing frameworks and newly-emerging channels of communication conduce to advances in the integration infrastructures/frameworks for the applications.

3.1.3. Benefits of Enterprise Application Integration

Using EAI applications share data and process without changing the applications and their data structure which cost effective [6] EAI enables access to real-time synchronous and asynchronous data between different systems and applications. It also helps creation of more usable applications by increasing performance in the implementations of business processes, and provides data integrity, hence reducing the cost of development and maintenance of a new application since it enables applications to run independently from each other.

Integration with corporations providing more corporate and commercial infrastructure in the market enables flexibility helps the maximization of corporate profits by responding consumer needs faster because it is always easier and quicker to integrate existing applications rather than develop new ones, reducing the "time to market" span for the work done.

3.1.4. Types of Enterprise Application Integration

There are several ways to integrate applications. Each integration approach has its advantages onto others considering the criteria for integration. Although there are numerous approaches, it is more useful to classify them under four main groups [32].

File Transfer: Each application produces data which is to be shared, used and processed by other applications.

Shared Database: Each application stores the data which is to be relayed to the shared database.

Remote Procedure Invocation: Some procedures are displayed in each application in order to invoke the applications remotely and each application invokes those to run behavior and exchange data.

Messaging: Each application connects to a system of messaging, and exchange data and invokes behavior by using messages.

3.1.4.1. File Transfer

Applications used by an enterprise are generally built with/in different languages and platforms. There are typically numerous software dealing with various tasks assigned by the enterprise. What leads to this multi-polarity in software can be summed up in a variety of reasons: The enterprise makes package purchases that are developed outside, the technology advances at such a rate that leads to discrepancies between the systems built at different times and there is the human factor: systems are developed by different people whose capabilities and choices affect the overall approaches to the development of new applications.

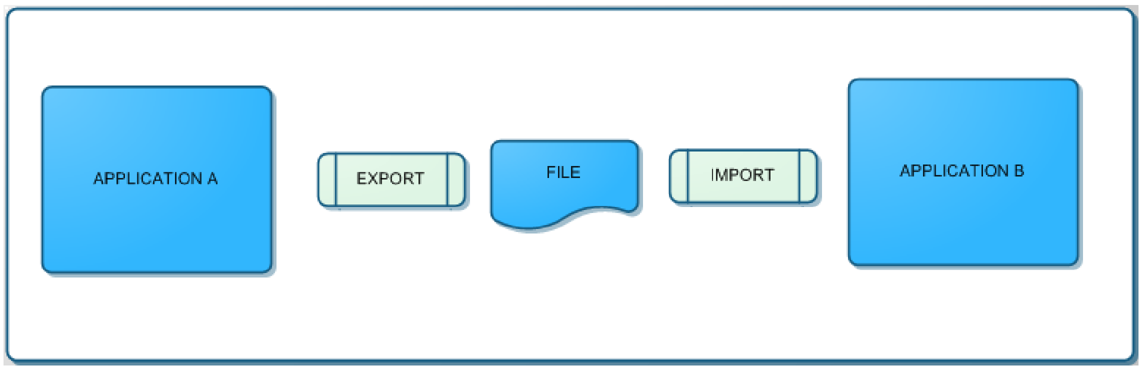


Figure3.2. Integration by File Transfer [32]

Various applications are written in various languages and established according to various platforms, along with the expectations that the application is developed upon. Establishing a link between this kind of applications require a profound understanding of how those applications interact together on both a business level and a technical level. To be able to grow this understanding, at first one need to know how the applications work, yet this has to be done in a minimal way so as to eschew confusion. After analyzing how the applications work, there needs to be a medium for the applications to interact with each other, which can be achieved through a common data transfer regardless of the language and the platform, yet seeming natural to the applications in question. This process needs to be attained with the minimum number of software and hardware, along with the applications that are already used by the enterprise. One of those common applications is the universal storage mechanism present in any enterprise. The basic approach would be the integration of the applications that utilize files. Each application has to produce files that have the information which is processed by the other applications. The transformation of those files into various formats is done by the integrators. The files are produced at a routine fashion depending on the business trajectory, yet the essentiality of the whole point lies beneath the format used for the files. Generally the out coming data out of an application is not exactly the same with the data required by the other application, thus making the processing phase crucial at this point, yet in order for the data to be processed, the data should be developed in a way that the processing tools could work on it as well. Considering all the procedure, standard file formats have had to adapt to differentiating needs. Mainframe systems mostly use data feeds that are based on the file system formats of COBOL. UNIX systems, however, use text based ones. There has recently been a trend to use XML.

An advantage of using files is that integrators do not have to know anything about the internals of an application. The file is usually provided by a team working on the application. The content and the format of the file are determined according to the needs, if a package is used, however, options are limited. The integrators make the transformations needed by other applications, or they just let the receiving applications decide how to handle and read the file. Therefore differentiating applications are separated from each other and each application is capable of conducting internal changes in a free fashion without manipulating other applications as long as they relay the same data in files the same format. Hence, the files come to be the interface of each application. [32]

File Transfer is a simple process partly because of the fact that there is no requirement for extra tools or integration packages, yet meaning that developers have to do the most of the work by themselves. The file names should be unique, so the writer of them has to have a strategy. Also, an application among others should be selected to know when a file is old and no longer needed, and to execute the deletion of those old files. A locking mechanism or a timing adjustment should be introduced, too, in order to prevent a reading of a file while it is being written by another application. Unless all applications are able to access to the same disk, an application has to act to carry a file from one disk to another. The longer the process of transferring files takes, the more probable that there will be problems because of inconsistencies.

3.1.4.2. Shared Database

Sharing data between applications is possible by file transfer, yet it may lack proper timing, which is quintessential in integration. If the alterations made do not reach quickly to other applications, the incorrect timing might lead to disruption in applications. To have the most up-to-date data is the key point because it is seen as reliable and error-free.

Updated are helpful to handle inconsistencies. The more synchronization in the system means the least possible errors to clear up. However, despite the updates, there is still a possibility of problems.

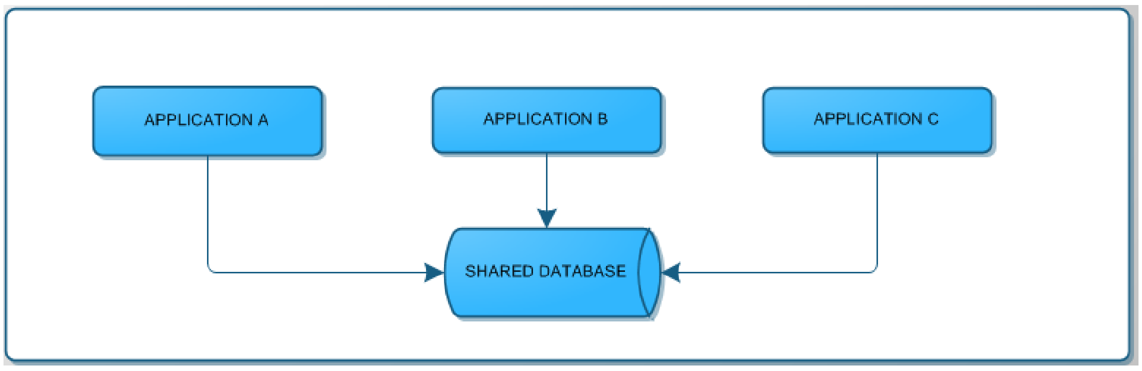


Figure 3.3. Integration by Shared Database [32]

In file transfer data format may not be well kept thoroughly, causing problems. In fact, in the integration process, many of the problems result from the fact that data are seen differently by different applications. Slight business issues may have a profound effect. Thus central data storage is indispensable for applications so that they can access to any shared data from a single spot. Hence, integrating applications by getting them to share their data in a single database is useful and practical.

Only if all applications are connected to the same database that it is possible to accept that the data is consistent all the time. If a single piece of data is updated from different sources at the same time, transaction management systems can deal with it easily and as the temporal gaps between the updates are tiny; errors are much easier to spot and to fix.

SQL-based relational databases have been commonly used and this has made shared database much simpler. Nearly all application development platforms are able to collaborate with SQL thus there is often no need to be concerned about various file formats and if everyone uses the same database, there cannot be problems in semantic dissonance. Thus, dealing with any possible problems becomes much easier before the software goes there may still be problems, it is always much easier to solve them before the software goes live and start to fetch huge chunks of inconsistent data. [32]

If the shared database is used by multiple applications and these applications try to read and to alter the same data, this may well cause conflicts and deadlocks because each application, while it is accessing a particular data, keeps other applications away from using the same data. Thus, if applications are present across multiple computers, it should be ensured that the data from the database is obtained locally yet this complicates the decision on which computer to store all the data. Hence a database that is dispersed among computers can develop locking errors and of course a failure in operability and performance.

3.1.4.3. Remote Procedure Invocation

An essential part of integration across applications is the sharing of data made possible by File Transfer and Shared Database, yet this may not be sufficient. A change in a data might mean that change is also to be done in other applications, too. For example, changing an address might seem like a minor alteration, yet it may spark legal processes to consider other rules in different legal jurisdiction.

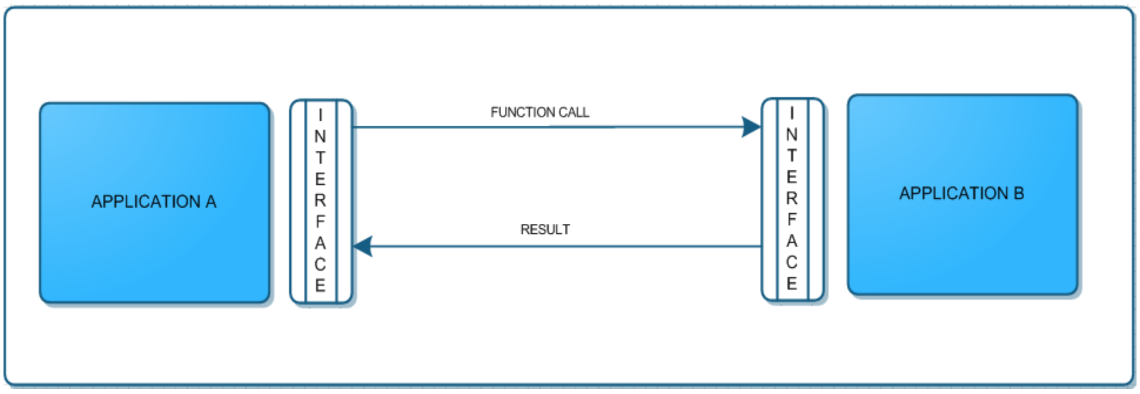


Figure 3.4. Integration by Remote Procedure Invocation

For an application to invoke such processes in others entails it to know the internals of other applications, which reflects the classic issues in the design of the applications. Encapsulation – one of the most powerful structuring control systems in application design – makes it possible for applications to store their data in a closed way with the help of a function called interface. Thus, when the data is changed, it is feasible to intercept changes in it to perform actions the applications have to do . Shared data – sustaining a huge, encapsulated data schema- makes this more complicated. For File transfer, it lets an application to respond to changes while processing a file, yet in the end, the process takes much more time. Changes in an application might trigger an alteration in the database, which might render a ripple effect across all applications.

Hence, the systems that use Shared Database are not flexible to changes in the database, which basically come to mean that application development may not be shaped according to the demands of the business.

Wrapping the data facilitates solving semantic dissonances. Multiple interfaces might be appointed to the same data for applications to read it in their own way. Even updates might make use of interfaces, enabling numerous points of view rather than associational views. Yet, integrators do not often include transformation components, forcing the each application to intermediate the interfaces with their neighbors. Remote Procedure Invocation is well-known to software developers as they are accustomed to procedure calls.

3.1.4.4. Messaging

File Transfer and Shared Database allow applications to sharing of data but not functionality. Remote Procedure Invocation provides functionality yet it pairs the application so tightly in the process. However; Remote Procedure Invocation looks like a practical choice but applying a model for a specific application onto the integration of applications has its drawbacks one of which is the problems resulting from distributed development and in spite of the fact that remote procedure calls may resemble local calls, they behave differently and are much slower, which increases the possibility of failures. Messaging enables packet transferring much more often, in a quick, reliable, asynchronous way and in formats that can be altered. [32]

Transferring data by messaging asynchronously can deal with the problems which occur in distributed systems. Both systems do not have to be ready simultaneously. Moreover, developers admit that with asynch communication , a remote application is slower thus promoting designs that are prone to do lots of local work and relatively less re-more work.

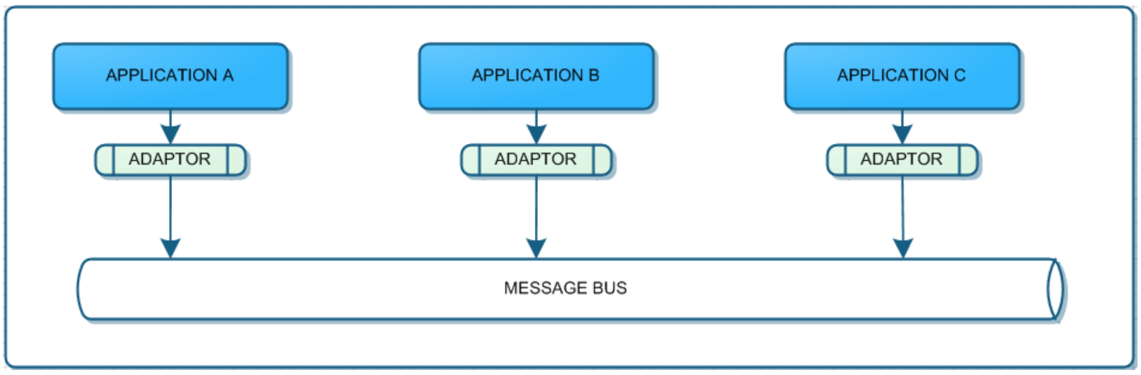


Figure 3.5. Integration by Messaging [32]

If File Transfer is used, messaging enables decompounding as well, in which case the forms of the messages can be altered on the way unknown to the sender and the receiver. Decompounding enables sending messages to numerous receivers and picking one of possible receiver.

Sending tiny messages in a more often fashion enables applications to cooperate behaviorally while sharing data. If a process needs to be launched once an insurance claim is received it is turned into a message right away as soon as a single claim comes in. Request of information and reply are made quickly. This cooperation is not fast as it is in Remote Procedure Invocation, yet the caller does lose any time by stopping as the message is in process and a reply reaches the destination.

Being able to send as many messages as possible in Messaging minimizes the inconsistencies that File Transfer fails to cope with. However; there still can be some lag problems in the systems not being updated simultaneously. People in the software business do not know much about the asynchronous design, thus resulting in different rules and techniques.

Changing the format of messages enables much more room for decompounding for applications, which is not the case in Remote Procedure Invocation and File Transfer.

3.1.5. Topologies for Enterprise Application Integration

EAIs are middleware software systems that help different systems communicate with each other. When examined the application schema of these systems, it is observable that there are three basic topologies. In my thesis, I will touch upon the advantages and disadvantages of these three topologies: Point-to-point topology which was firstly used and seen as the most basic solution, hub-and-spoke topology administered from a system that is center of integrations and last but not least, bus topology which is the architecture of enterprise service bus.

3.1.5.1. Point to Point Topology

Most of integration projects are the results of the need for communication between two systems. The most practical way of providing this communication is to utilize *Point-to-Point Connection.* In point-to-point connection only one receiver get one particular message providing that the system knows where that particular message to be delivered. The sending system usually has to transform the message into the format which could be understood by the receiving point.

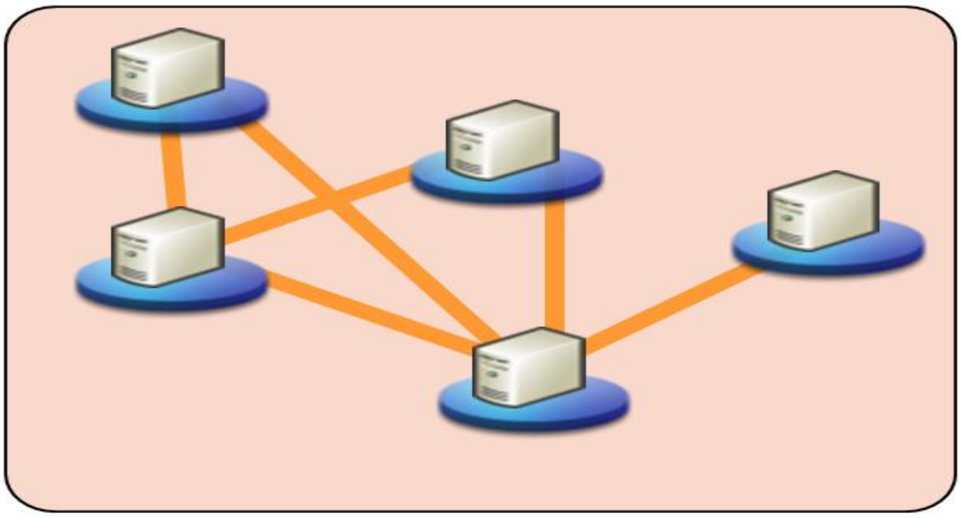


Figure 3.6. Point To Point Topology

In point-to-point connections, the addresses of all nodes or points that need to be linked are determined by the system. If there are changes in target addresses or protocol details, an update is required for the systems. Furthermore, if the integration network grows larger and at the same time changes become recurrent, it is likely that operational cost of maintaining system adopting this approach becomes notable.

In most of the integration projects, data is expected to be transformed between the source system and the target system. Moreover, developers sometimes may want to make use of some conditional logic while customizing message routing. In point-to-point connections, a duplication of aforementioned logic is present on each server in need of transformation and routing yet writing a duplication code might be costly, hard to maintain and to test [15].

Advantages

- Integration is the simplest of all and tightly bound
- Enables better integration with small number of systems.

Disadvantages

- There is limited flexibility and constant need for updates.
- The more integration points to take care of the more complex it gets.

3.1.5.2. Hub and Spoke Topology

In Hub-and-Spoke topology, there is a centralized broker which is called a hub and there are adapters, namely, spokes which enable applications to connect the Hub and they convert the formats of the application data to that of the Hub recognizes, or vice versa. The Hub deals with all messages, their transformation processes into the format that destination application understands and the routing. Spokes get data from the origin application as relay messages to the Hub, then the Hub passes those messages to a subscribing adapter and it send those over to the target application.

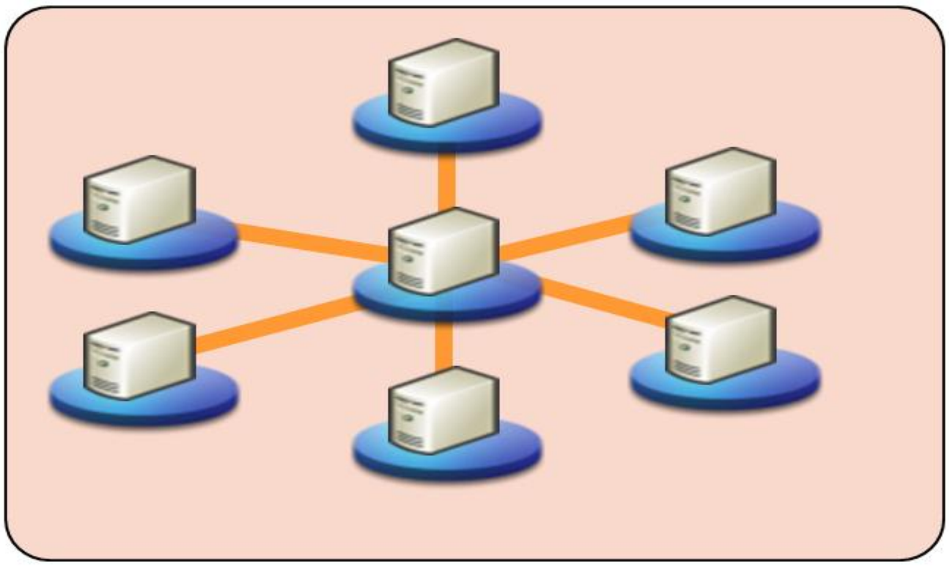


Figure 3.7. Hub and Spoke Topology

To create a central location for control, hub topology is very helpful and the source sends the messages to the central hub. Hub topology is very effective provided that business events are not dependent and if a single vendor provides the Message Oriented Middleware (MOM). The source application here forms a message in a particular format and the hub re-forms and sends it to the spokes linked to the hub [16].

Advantages

- Enables integrations via central management.
- Less complexity compared to point to point.
- Business process is controlled and mapping in data layer is provided
- There is more scalability.

Disadvantages

- All system is susceptible to single point of failure.
- There is limited scalability for technologic infrastructure
- The available hubs cannot generally deal with the incoming transaction duties from other sources except the middleware they work on.
- Integration processes with multiple sources and destinations are hard to manage.
- In need of a database, processing or routing bottlenecks crowd the hub since volumes grow and integration rules get more complex.

3.1.5.3. Bus Topology

Messages from source applications are put onto a system-wide logical software bus that other applications can access. That's why, bus topology is beneficial for relaying information to multiple destinations. Messages on the bus can be particularly subscribed by multiple applications and the data relay may not have to pass through the central switching point, which is possible only in publish and subscribe middleware. The glitch of bottlenecks is, however, overcome by bus topology.

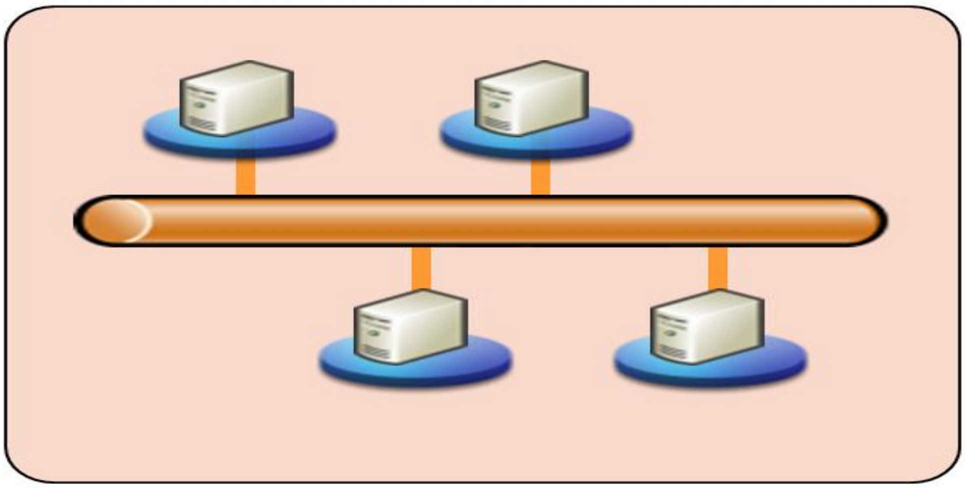


Figure 3.8. Bus Topology

A central messaging bus is utilized for the distribution of messages by bus architecture and the messages are published by applications to the bus using adapters. The message bus takes these messages to the subscribing applications which contain adapters taking the messages and re-forming them into the required format [33].

Advantages

- Enables integration of loosely coupled services.
- Enables infrastructure for shared communication
- Service Meditation

Disadvantages

- It is hard to control all messages on bus
- It is hard to adapt systems to loosely coupled services
- Latency period is increased compared to point-to-point integrations

3.2. Enterprise Service Bus

Before touching upon the definition of Enterprise Service Bus (ESB), it is essential to clarify what Service Oriented Architecture (SOA) is and what features it provides since ESB is the message-based integration architecture containing SOA features and supporting its infrastructure. Agility, Flexibility and Reusability, basic features of SOA, form the foundation of the advantages provided by the ESB architecture. These three are the features that are targeted and benefitted as much as possible by the infrastructure that is the focal point in my thesis.

SOA can also be seen as an architectural format that backs weakly coupled services in providing flexibility in businesses in a way that enables interoperability in an multiple-technology environment. SOA is comprised of a complex group of business-aligned services enabling the actualization of adaptable and customizable business processes by utilizing interface-based service descriptions [7].

The aim of getting SOA to deepen IS and business activity, and to ameliorate IT-business alignment in multi-atmosphere business conditions is not very explicit in the definition. SOA differs from other ITs in that it accentuates more on IS agility thus ameliorating business agility. The closer the link between IT and business, the more quickly an organization can act to alter IS applications according to business needs.

SOA provides methods for systems development and integration where systems group functionality around business processes and package these as an interoperable service [8]. An organization can make use of these services by re-using them or these might also be commercially on the market. Thus, SOA separates functions into distinct units, or services, which developers make accessible over a network in order that users can combine and reuse them in the production of business applications. [9]Between these services, there is always a strong communication which consists of data exchange, and enables coordination of an activity processed in two or more services. Data transfer between reusable services and cross domains is easily achieved. Reusable services reduce integration costs in SOA architecture and facilitate the integration of end points to the system. Many end points in SOA architecture contain single service availability for use. That’s why each implemented service is designed and developed independent of business flows, other enterprises or technologies. Services which are on SOA architecture and can be called from more than one place, by increasing the reusability, become available for the use of multiple external systems at the same time, and enable updating of the changes on the whole system with a single move when there is a need for change.

In order to elaborate more, it is useful to define Enterprise Service Bus first. Enterprise Service Bus (ESB) is a platform that gather messaging, web services, data transformation and intelligent routing in a way that links numerous different applications across an organization and its partners and coordinate them while keeping transactional integrity. It makes use of the features provided by Service Oriented

Architecture (SOA), Enterprise Application Integration (EAI), Business-to-Business (B2B), and web services, thus making itself an integrated platform enabling essential interaction and communication services that complicated software applications need via an event driven and standards-based messaging engine, or bus, built with middleware infrastructure product technologies [10]. By insulating the link established among a service and a transport medium, it is utilized to realize the needs of service-oriented architecture (SOA) [11].

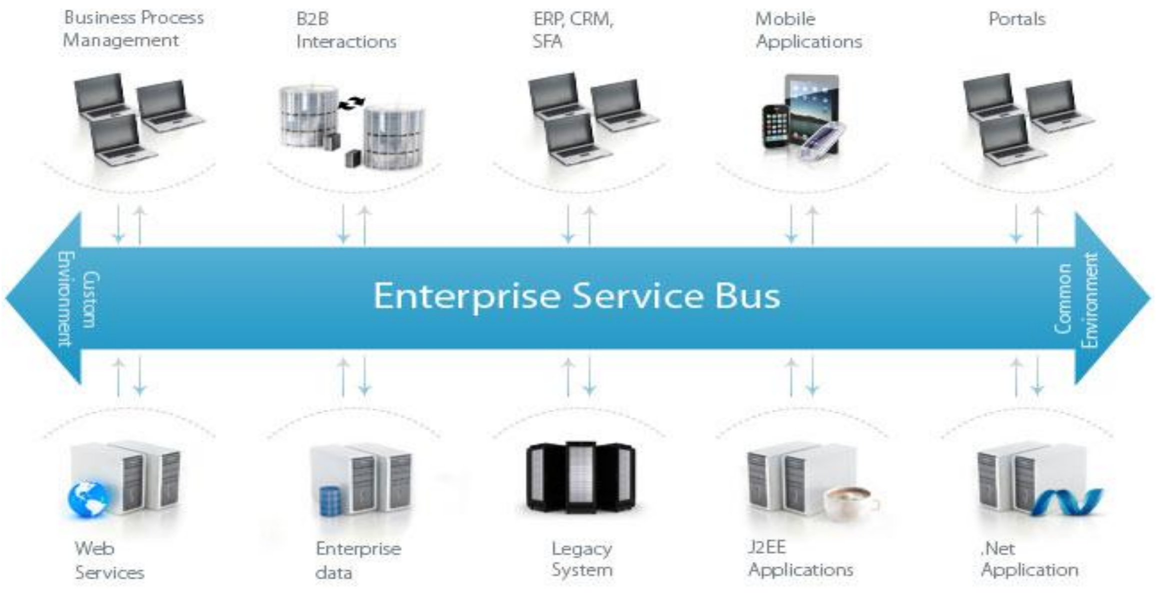


Figure 3.9. Overview of Enterprise Service Bus [34]

In this way, interoperability among diverse situations is achieved with the help of using a service-oriented model. Despite being thought to be linked to concepts like integration and mediation, ESB, in a way, merges integration and application server product categories. One ground-braking features of ESB is that it is able to virtualize services. A service container of ESB holds a service and isolates it from its protocols, methods of invocation, method exchange patterns, quality of service needs and other infrastructure concerns.

Furthermore, ESB is able to supply a kind of abstract stratum for an established enterprise messaging system enabling integrators to employ the advantages of messaging without writing down any codes. An ESB is based on basic functions parted as primary parts with distributed deployment and collaboration as opposed to the techniques utilized by traditional enterprise application integration (EAI). In addition, flexibility and multiple transport media capability are supported by structural constituents of ESB. Basic features that need to be provided by ESB architecture are [35]:

- Supporting the ability to invoke services
- Employing routing through dynamic mechanisms
- Sustaining service mediation
- Supporting messaging and some other features which might also be beneficial.
- Being weakly coupled and changing to event needs
- Supporting WS-* standards
- Sustaining quality in service management
- Providing process orchestration

With the features it provides, ESBs are developed versions of message-based EAI systems. Enabled by SOA architecture, they aim to minimize integration difficulties and to reduce improving costs. The features above are the ones which will facilitate the integration of complex business processes that need solutions in different domains. The need for ESBs has arisen in time to tackle with the challenges in hub or point-to-point frameworks which are mentioned in integration typologies.

The main reason behind the ESB pattern is, however, to establish a framework which enables developing service-focused applications that are capable of overcoming challenges in the early phases in the integration process. This is possible by concentrating the logic from each group of end-points into a centralized stratum, a connection per service. ESB pattern is different from EAI which was founded on a centralized stratum that it focuses on dynamic execution. Since business requirements change in time, software has to change according to those requirements, thus making extensibility very essential. The rationale behind the ESB pattern is that it could provide effective changes according to business needs in a fashion which makes a focused, loosely-coupled, dynamic layer available for the management of integrations. In this way, the software is a lot easy to handle and maintain, enabling the business to increase value by reducing the operating costs and the time needed for fixes [12].

3.3. Software Products for Enterprise Application Integration

Enterprise Application Integration products are widely used in information technology. Some parts of these are open-source and free products and may be improved. Mostly used ones of these products are, BizTalk Server, Sonic ESB, Mule ESB and Oracle Enterprise Service Bus. In my thesis, i worked with BizTalk server and a toolkit of it Enterprise Service Bus Toolkit.

3.3.1. BizTalk Server

BizTalk Server, however, is an integration server developed for the integration of corporate applications. It enables communication between end-point applications in multiple platforms and works fully integrated with other Microsoft products, despite without providing hardware infrastructure. According to Microsoft, BizTalk is the number-one integration solution and value leader worldwide. These customers trust BizTalk for solutions such as payment processing, supply chain management, business-to-business interactions, real-time decision making, and reporting [13].

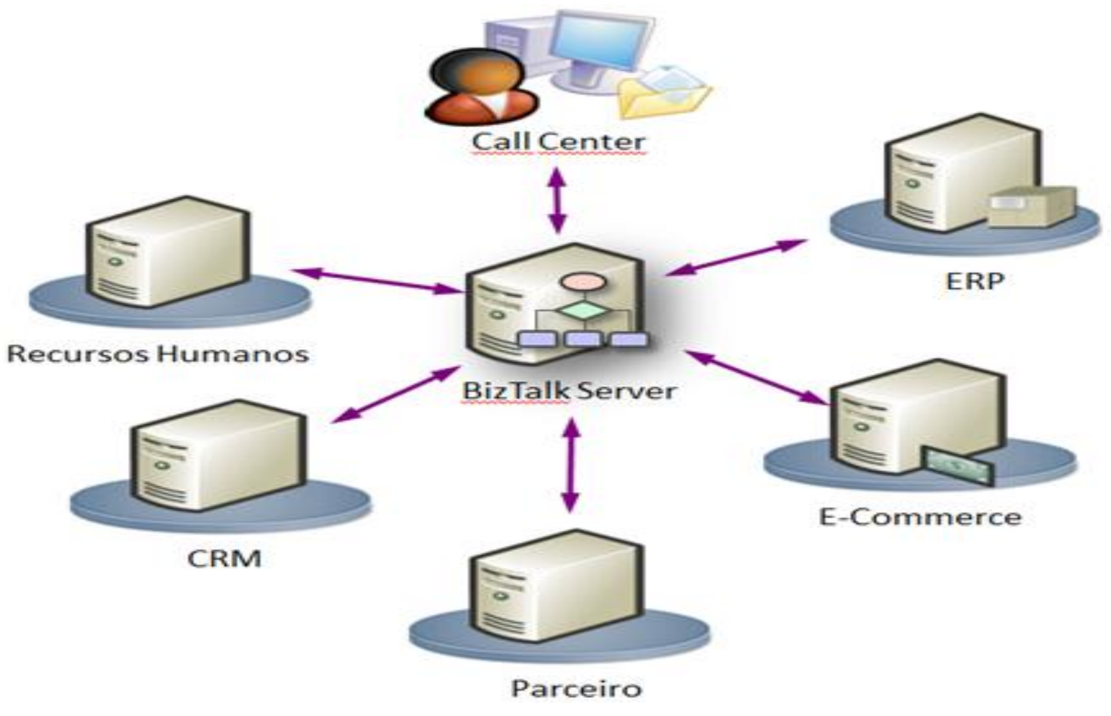


Figure 3.10. BizTalk [36]

With its first versions, BizTalk has adopted Hub & Spoke architecture and integrations it provides have the characteristics of this architecture. Whereas Hub &Spoke architecture has the aforementioned advantages, there are some drawbacks emerge in time. BizTalk server acts as a management tool on the central server and makes integration flow possible between multiple points.

BizTalk is a Microsoft product and works in an integrated way with the other products the company provides. It enables development and service of integration along with operating systems, databases and development tools; however, it is not possible to use it on its own. It is dependent upon Windows Server operating system and SQL Server Database. BizTalk also allows developing and running of orchestrations for the management of business processes and provides solutions to connect multiple different platforms and systems such as SAP. BizTalk, however, is not a product that can be configured in runtime, that's why it cannot fully support some of SOA features. All configuration is made in development time so any change in business process needs a development, build and deployment process again.

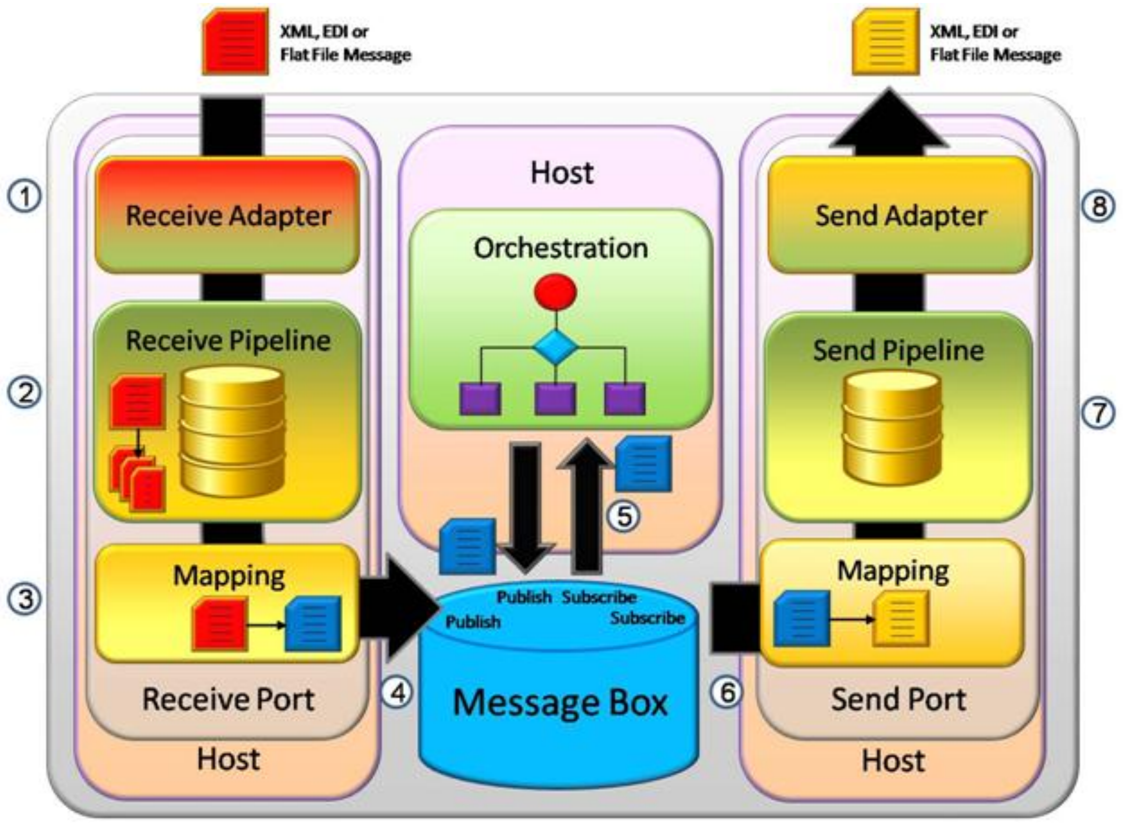


Figure 3.11. BizTalk Server Architecture [37]

It can be seen in Figure 3.11, all configuration in BizTalk determined in development process. When it is examined a message flow during BizTalk, the steps in message flow are [37],

- A message included in a Receive Location that is statically configured in BizTalk. This location can be FTP Server directory, Web Service URL, Database table or etc.
- Incoming message pass through a appropriate pipeline which process incoming message to its schema defined in BizTalk
- Transformation process executed with a Map over incoming Message.
- Message is inserted to Message Box Database
- Business Logic is process if it is developed in design time in Orchestration
- Incoming message is routed by a send pipeline with appropriate Send Adaptor to statically configured end point location

In the process flow it is seen that, service oriented architecture capabilities is missing in BizTalk Server message processing flow. Hence, Microsoft released a product over BizTalk which provides main SOA advantages in EAI.

3.3.2. Enterprise Service Bus Toolkit

ESB Toolkit is, on the other hand, is a compound of tools implementing ESB architecture of Microsoft and developed on a BizTalk Server product which provides customers with the advantages that ESB brings about. According to Microsoft, ESB Toolkit is a *collection of tools and libraries that extend BizTalk Server 2010 capabilities of supporting a loosely coupled and dynamic messaging architecture* [14].

ESB Toolkit works as an intermediary between services and their consumers and enables fast mediation between them along with providing utmost adaptability at run time. BizTalk ESB Toolkit 2.1 reduces complexities in the composition of service endpoints and management of interactions between services.

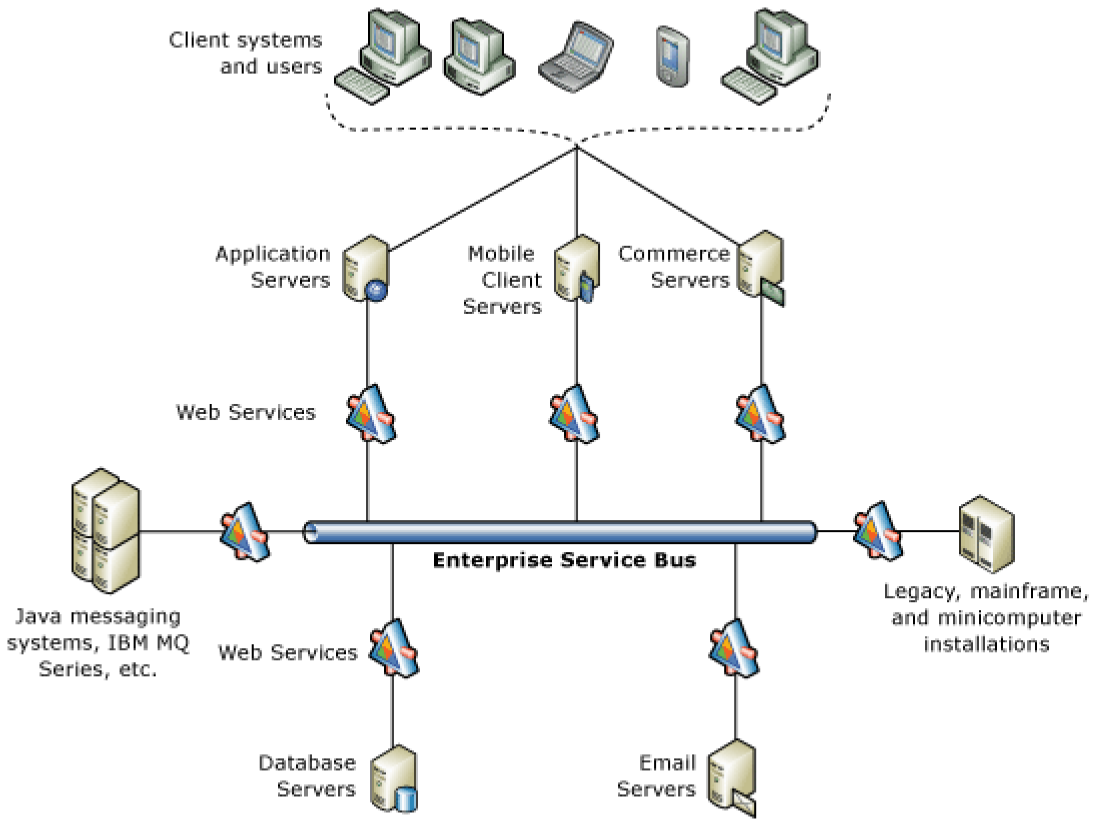


Figure 3.12. Microsoft ESB Toolkit [38]

As mentioned earlier in ESB definition, SOA infrastructure is one of the basic features provided for ESB. Basic features of ESB and opportunities it provides are as follows.

- **Endpoint run-time discovery and virtualization:** Virtualization of end-points and actualization in run-time.
- **Loosely coupled service composition**: Enabling access to servers from every location and dynamic use of these servers which do not have direct connection.
- **Dynamic message transformation and translation.** Dynamic transforming of messages and their interpretation.
- **Dynamic routing.** Run-time, content-based, itinerary-based, or context-based message routing.
- **Extensibility.** Provides multiple extensibility points to extend functionality for endpoint discovery, message routing, and additional BizTalk Server adapters for run time and design time [14].

How ESB Toolkit achieves to provide SOA advantages? Considering into architecture of ESB Toolkit, there is a Abstraction layer over BizTalk Services. In case of study section, it is declared on architecture which services are used and improved.

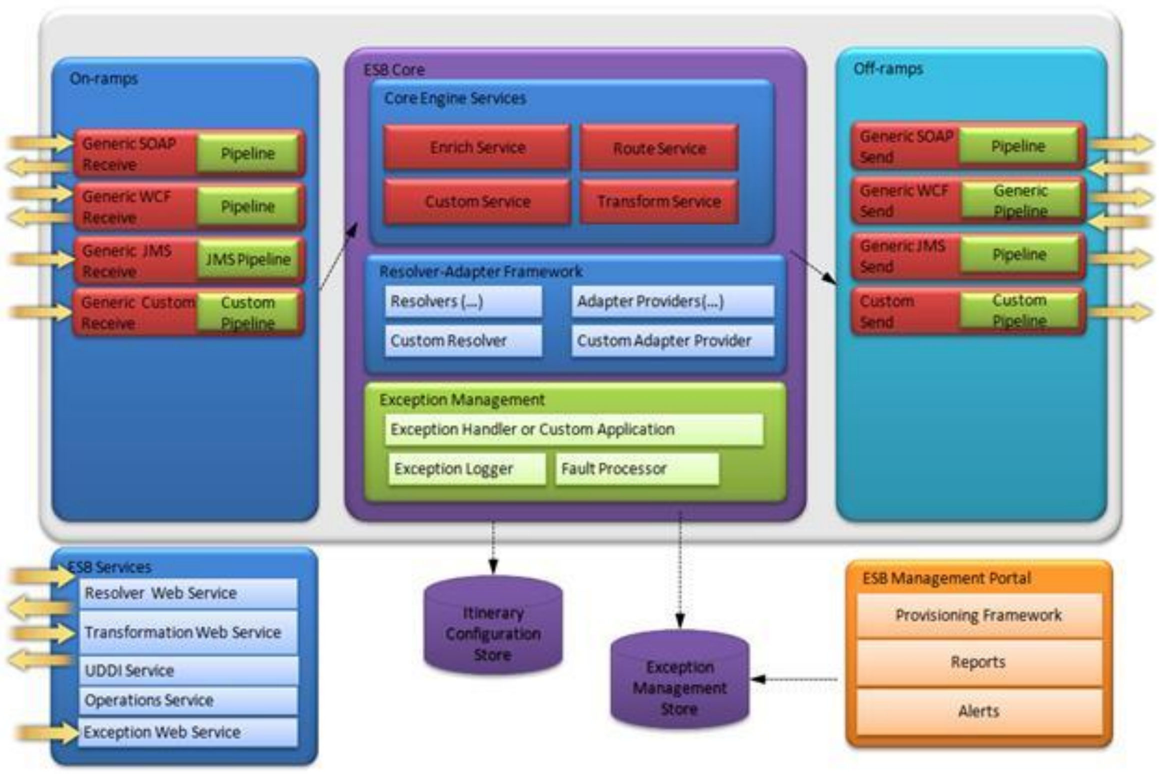


Figure 3.13. ESB Toolkit Architecture [38]

A generic service of ESB Toolkit provides dynamic resolution of maps and end point to route. Each service is a key for SOA infrastructure. ESB Guideline explaining frameworks and services as,

The Resolver and Adapter Provider Framework provide a comprehensive, pluggable architecture for dynamically resolving endpoint information and BizTalk Server 2010 map types. It uses extensible components, which allow developers to change the behavior to suit their own requirements and extend the mechanism to support alternative resolution and routing methods [23].

- Resolver service: This service allows external consumer programs to leverage the resolution mechanism. The Resolver service can be used to abstract service registry access and make it broadly available in a heterogeneous environment.
- On-ramp service: This service provides a means for Web service consumers to send messages to the ESB. Web service SOAP headers become message context

properties as the message passes through a context setting component in a receive pipeline.

- Transformation service: This service allows non-BizTalk applications to access and leverage the BizTalk transformation engine. Specifically, it allows access to all Web service consumers including those not running on the Microsoft platform. There, we use transformation engine to run BizTalk maps, through different integrations. So, the service provides reusable service via SOA infrastructure.
- Exception Management service: By publishing the fault schema using the default BizTalk schema publishing mechanism, this service enables consumers to submit messages so that non-BizTalk can participate in the ESB exception management scheme. This Service has both BizTalk supported solution and a Web Service implementation that can be used from different applications from different platforms.
- BizTalk Operations service: This service returns information about the current state of BizTalk artifacts. These services are available to implement a service-oriented solution through ESB Toolkit. ESB toolkit is the alternative way to define post-production or runtime configurations which were not able to be done with BizTalk Server. This ability supports the advantages of Service Oriented Architecture that are defined above.

These ESB Toolkit services are used within a integration in ESB and message flow is processed by these services. In ESB Toolkit message flow configurations can be gather runtime by Resolver and Adaptor.

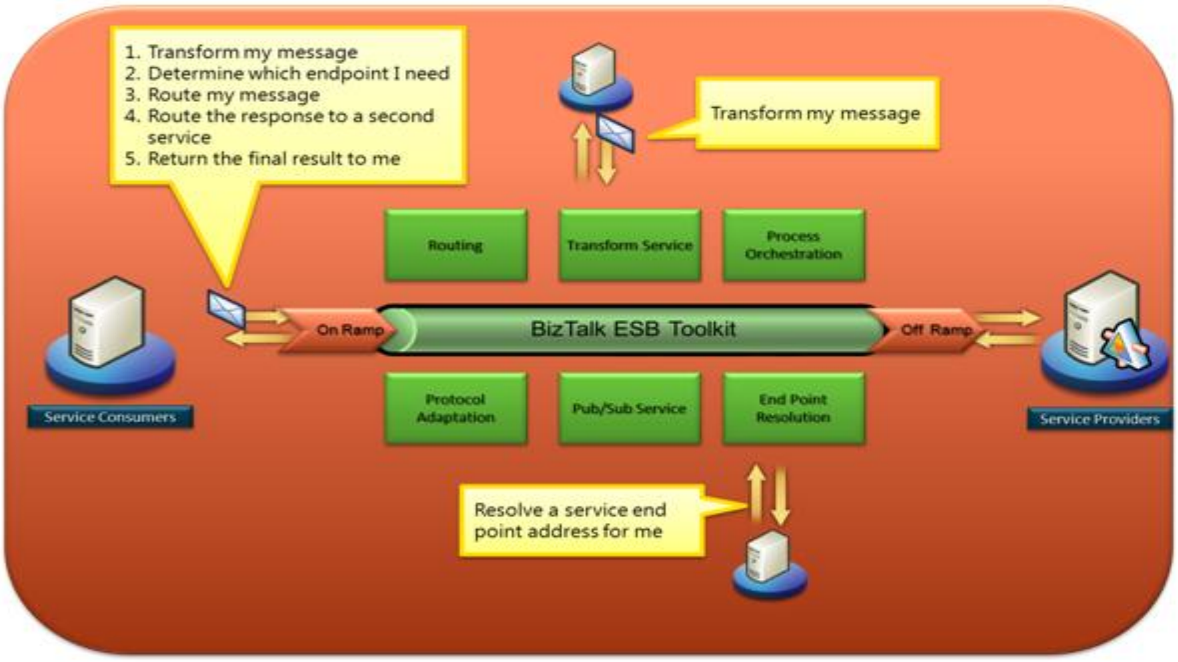


Figure 3.14. ESB Toolkit Sample Message Flow

Message processing in ESB Toolkit is provided by service consuming in run-time. Services uses resolver and adaptor framework which explained above. Compared to BizTalk Server message processing, Resolver and Adaptor Framework give ability to resolve configuration of services during run-time.

3.4. Container Logistics Business Process

Container logistics management is a supply chain management component that is used to meet customer demands through the planning, control and implementation of the effective movement and storage of related information, goods and services from origin to destination. Container logistics management helps companies reduce expenses and enhance customer service [24]. It includes lots of vendors and partner.

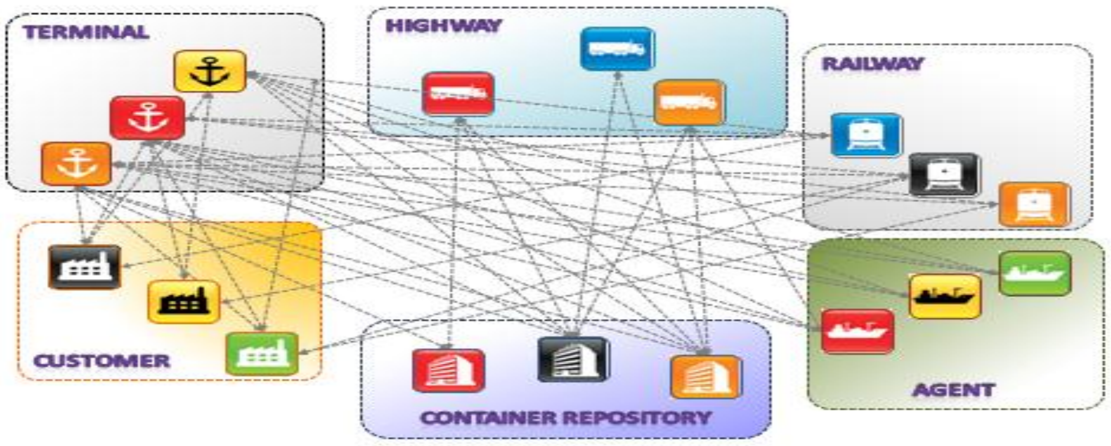


Figure 3.15. Container Logistics Business Process [19]

There are multiple dimensions how to examine and categorize container logistics which primarily focuses on transportation and can roughly be divided into sub-categories three of which will be studied in this thesis. These are: Road transportation, rail transportation and maritime transportation.

In container logistics, the most crucial point of transportation is the shipping container. All products from different sectors are transported in containers. As will be used frequently in the thesis, some explanations are useful before delving into the subject.

- Container : Container is a single rigid receptacle without wheels that is used for the transport of goods

Figure 3.16. Container group For Arkas Logistics

- Road transportation: Transportation of containers via road systems in logistical business processes. Depots play an important role in this kind of transportation.
- Rail transportation: In rail transportation, containers are transported by freight trains. Despite being cheaper and more sufficient, rail transportation may not always be useful in direct, to-the-point shipments, thus making road transportation an essential step in the process.
- Maritime transportation: Maritime transportation is the way of transporting containers with ships and this usually takes place intercontinental, hence, ports are essential parts of this method of transportation. Containers transported via road are brought to ports, transported onto relevant vessels. However, containers dismounted off the ships and related checks are, too, considered within this process.
- Agency: Agencies are establishments that follow up the business, establish communication between depots, ports and ship owners, and manage the transportation processes in accordance with the demands from companies. Container logistics in agencies play a crucial role in the process and are a part of the integration processes.

CHAPTER 4

METHODOLOGY

In this part, it will be elaborated on this thesis and will be explained the business domains and processes within which some studies have been done and will be continued with a business strategy.

After explaining the advantages and disadvantages of the integration models and their development, it will be elaborated on how ESB architecture is applied in this field and on its application methodology. The most crucial factor in the process is the grouping and modeling of integrations in business processes. Reducing the number of integrations is the primary determinant for my thesis to reach its target.

The domain that is utilized was container logistics business processes. As there are numerous different business flows and platforms in container logistics business processes, the number of running integrations is relatively high and the processes functioning in each integration differ from each other. The fact that each integration represent a different process results from the usage of integration architecture.

- Lack of modeling in business flows
- lack of application of integrations which are designed to operate together
- Need for maintenance and for new integrations after the application of point-to-point integration system with BizTalk Server increases the complexity of a working system.

Another problem in container logistical processes which is within the domain of thesis is the high number of companies that are in cooperation. While the multiplicity of present integrations makes the process more complex, one of the parameters that render solutions harder to attain is the increase in the number of cooperating partner companies. The need to cooperate with different companies in the same process, changes in details within the business logic requires more and more integration, thus posing an obstacle to the simplification of the integration processes. That's why, different companies have their own particular technological infrastructure and cannot be expected to show the same traits similar to each other, yet companies may have to take part in a business process with various technological infrastructures upon which divergent operating systems, platforms or different applications are present. At this point, with the process taken into consideration with a larger general view, a need to form an infrastructure that can work with different technologies arises.

However, to establish this infrastructure, it is important to determine common integration points, to centralize processes and to make sure that integration infrastructure can be developed.

That's why, the methodology taken towards this goal is as follows.

- On the container logistical domain at which our company worked with Bimar, there were around 200 working integrations and in time this number increases, as aforementioned. These integrations take place mostly between DEPOT, AGENCY and PORT.
- By analyzing the business processes for these integrations, then forming a model by generalizing the processes on swim lane diagrams, present integrations are demonstrated on these models. Below there is a sample of business process analysis. (See Appendix 1-9 for the rest of business processes analysis).

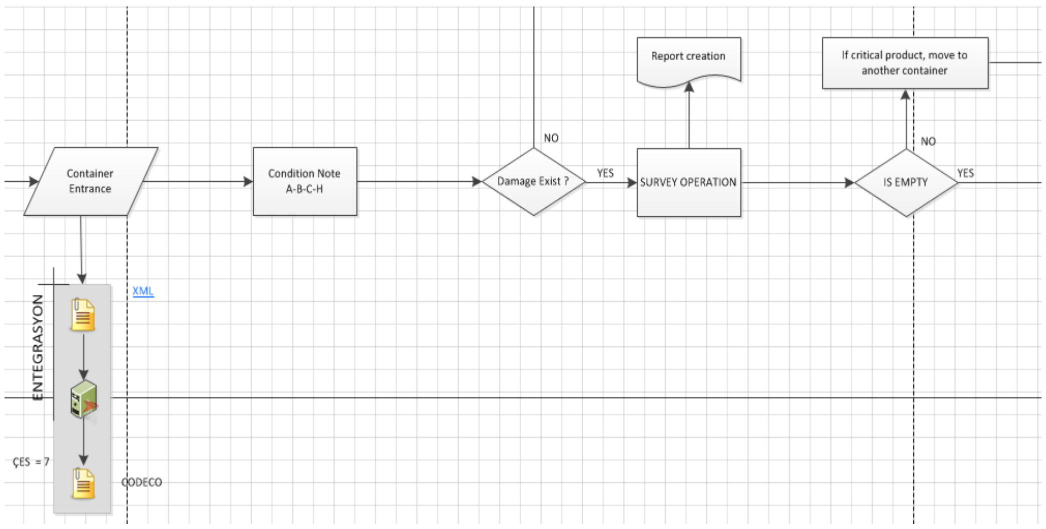


Figure 4.1. Containers Logistic Depot Business Process Analysis

- While modeling, integrations are divided into sub-groups in terms of business logic they work with and of input/output data types.

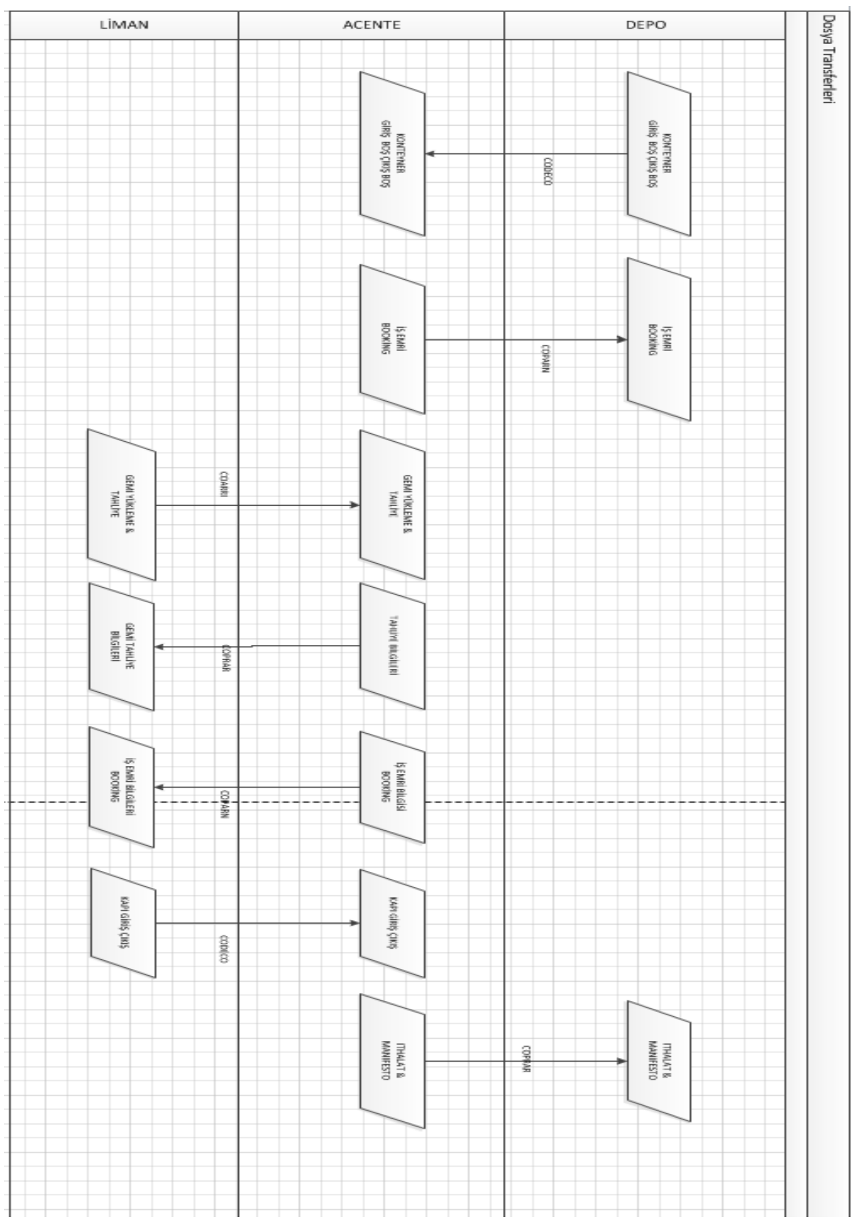


Figure 4.2. File groups between Depot, Agency and Port

- In order for business processes to work sufficiently, the incoming and outgoing data pose an essential role. Data or information should be entrusted to those who must have knowledge of it for its necessary usage. If you go backwards from usage to the need, then you may discover what is needed in order to start, continue or complete a business process [19].
- By utilizing the Formal Concept Analysis method, the aim was to simplify the integration processes. The purpose of FCA, according to Ganter and Wille [20] is to support the user in analyzing and structuring a domain of interest. Such a

method allows us to automatically obtain similarity scores without relying on human domain expertise [21].

- FCA here is intended to be applied onto domain knowledge. Inputs and outputs of each business process are present in that domain. In this way similarities of input and output will be seen and an integration point will emerge.
- With FCA method, by clustering input/output or business processes, integration points have been determined within the business processes of which inputs and outputs are similar or bear a high amount of resemblance.

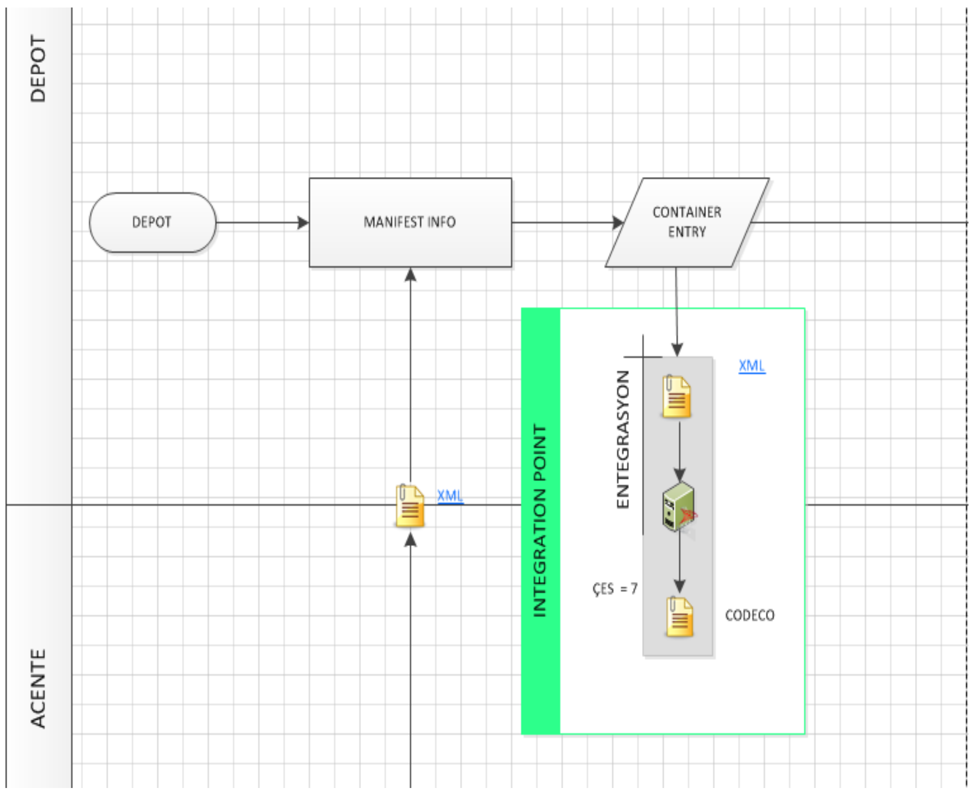


Figure 4.3. Integration Point between Depot and Agency

- Integration similarity method by data comparison is used in another article [30], that service data is declared as an entity and service entities compared to clarify a service model.
- For present integration working in determined spots and for similar integrations which can be added onto relevant business processes, the aim was merging for the grouped integrations.

- Reduction in the need of new and present integrations is targeted by means of grouped integrations in terms of processes and of integrations considered to be merged according to their input/output similarities.
- Merged integrations are processed by reusable custom services that provide integration tracking and exception handling in message flow.
- In [29] exception management is divided into 2 categories. First of them aims to handle exceptions while message acquired by system. The other one aims to handle exception inside system that message is being process according to business logic.
- Proposed solution in this thesis contains both of the categories. While incoming message is accepted by system it is tracked by exception handler and log services. Also, while message is being processed in business logic components it is being handled for possible exceptions.

Along with the methodology pursued, container logistical processes of road, sea and road transportation is analyzed, the processes for which the integrations are prepared are determined and on the analysis files formed integration points are indicated.

With the help of integration merging study, by implementing an infrastructure availability which grouped business processes can work with, present integrations are merged. The proposed infrastructure schema can be seen below,

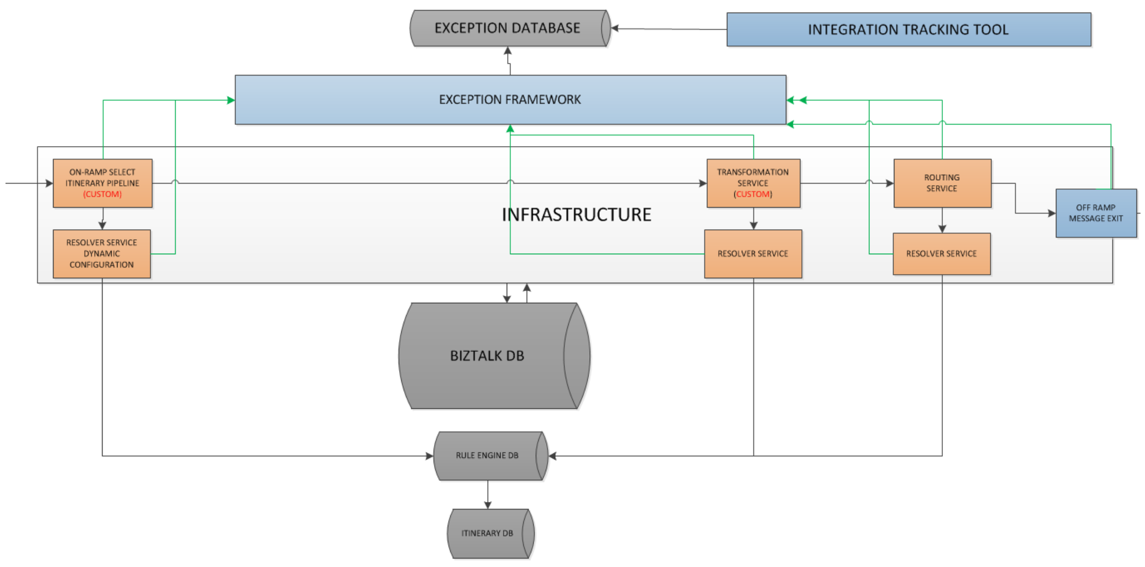


Figure 4.4. SOA Based Integration Infrastructure

Outcome of this study will be later explained in the next and in the case study with the benefits of the infrastructure framework written.

CHAPTER 5

PROPOSED SERVICE ORIENTED INTEGRATION OF LOGISTICS INFORMATION SYSTEMS

It has become a commonly used middleware infrastructure with the advantages of Enterprise Service Bus and with its support of SOA architecture. As IT managers develop their SOA plans they often come to the conclusion that infrastructure software is needed to fulfill their objectives for flexibility, robustness and control. The ESB has emerged as the pre-eminent form of SOA infrastructure software [22].

ESB provides a standart based integration infrastructure that combines messaging, transformaton, routing, exception handling and monitoring. Besides these , as explained in SOA advantages, ESB provides agile, flexible and combination of loosely coupled services.

Each service may be a processing part or end point in ESB system. These services give a flexible runtime environment for business process management. Each ESB vendor does not need to know the details of ESB services, so details can be gathered in runtime. Furthermore, each running service configuration can be modified in runtime and this configuration does not need a server or an application restart. This function gives a stimulus to agile business changes and approves the feature of SOA agility.

All the services in bus works as loosely coupled, easy to be configured in runtime, and reusable in ESB environment. Another critical issue about services in architecture is monitoring and exception handling. Service details, exception handling and exception analysis are needed to control business process flow over integrations.

Since integrations in operation are those which utilize P2P infrastructure, there is a need to adapt domain analysis and designated integration points to ESB architecture, and to form the software framework after the need for the domain. The problems of BizTalk integration infrastructure and the features of service-based ESB architecture that we present are designated below.

Since ESB Toolkit is a whole set of components working on BizTalk product, some of its services uses BizTalk artifects. During the development of software infrastructure, some of the present BizTalk services, ESB toolkit components were developed and new services are put into infrastructure use. Now, I will try to elucidate present-day BizTalk architecture, advantages of ESB compared to BizTalk and the steps to adapt the integration group operating in BizTalk architecture into the ESB toolkit infrastructure.

Arhcitecture of BizTalk relies on Hub & Spoke which is a point to point integration solution between cross platforms which can be seen in Figure 5.1. Through Hub & Spoke arhcitecture nature, its needs a centrailized system to supply integration connectivity.

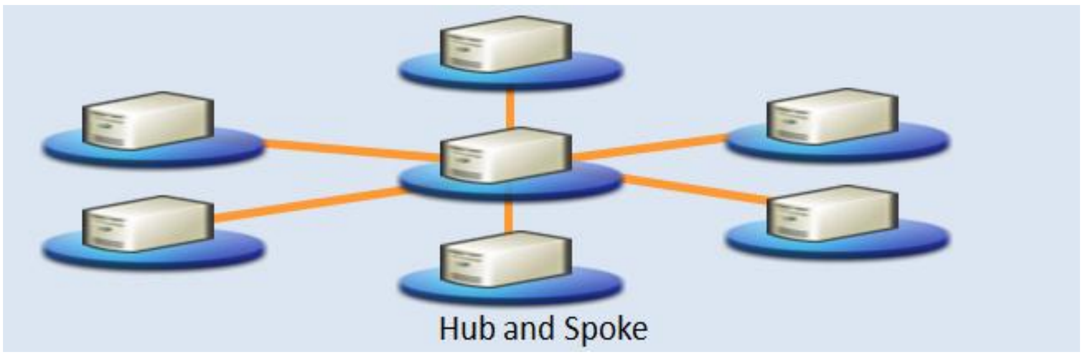


Figure 5.1. Hub and Spoke Architecture Diagram

In BizTalk environment, integration needs to be configured with certain parameters. Each end-to-end connectivity provided by BizTalk has a certain receive and end locations, a specific orchestration for business process management. Each integration has its own particular schemas and maps that can be processed through orchestrations if needed. Therefore, in case of a new integration demand, developers also have to prepare integration specific schemas, orchestrations, and business processes (orchestrations) and unique locations for receive and send. In Figure 5.2, static configuration for integrations is visualized.

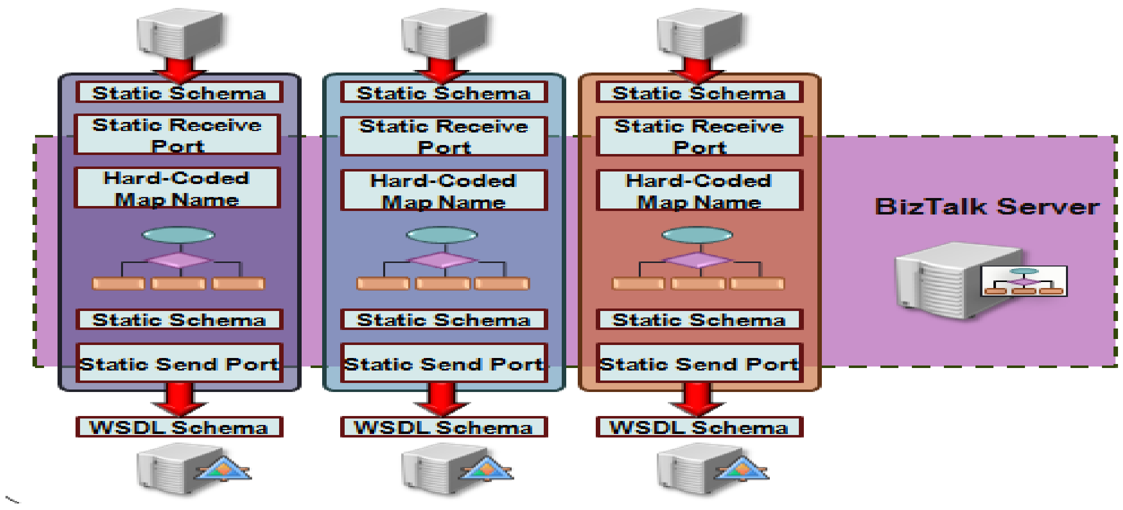


Figure 5.2. BizTalk Integration with static configuration

ESB Toolkit is a Microsoft Product is an architectural pattern for Microsoft Company to implement SOA based solutions. By adopting SOA based solutions over ESB Toolkit Microsoft provided to have advantages of using service-oriented architecture.

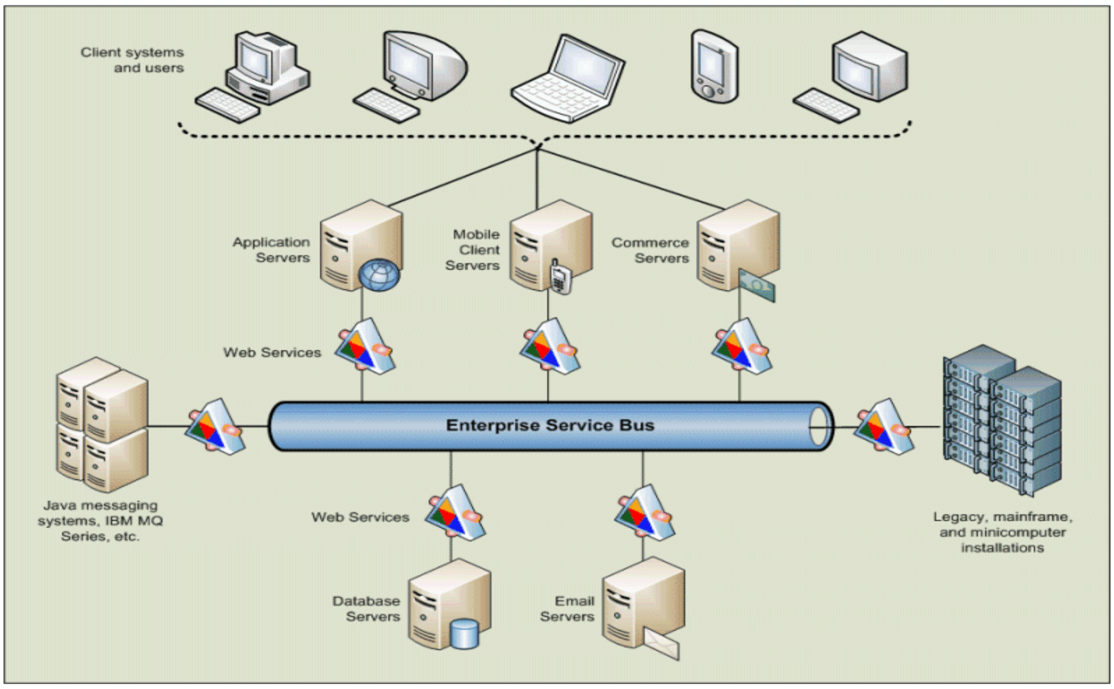


Figure 5.3. Enterprise Service Bus Toolkit [39]

SOA-based connectivity between cross platform applications supports flexible, agile environment with reusable services. ESB toolkit provides these advantages via abstraction of BizTalk components that are adequated to be implemented with SOA properties. ESB Toolkit has a list of reusable services, components, frameworks that BizTalk integration can be modified into an Enterprise Service Bus solution.

- Itinerary Creation

  Itinerary is the core component of ESB Toolkit architecture. It defines the message flow through Service Bus. The idea of itinerary is not to exclude business logic. For that, we have service composition capabilities using BizTalk orchestrations. The goal and appropriate use for an itinerary is a simpler series of steps, not a complete process. So, the first step is to create the itinerary for our message processing mechanism. This itinerary will include the service list that incoming message will be processed through.

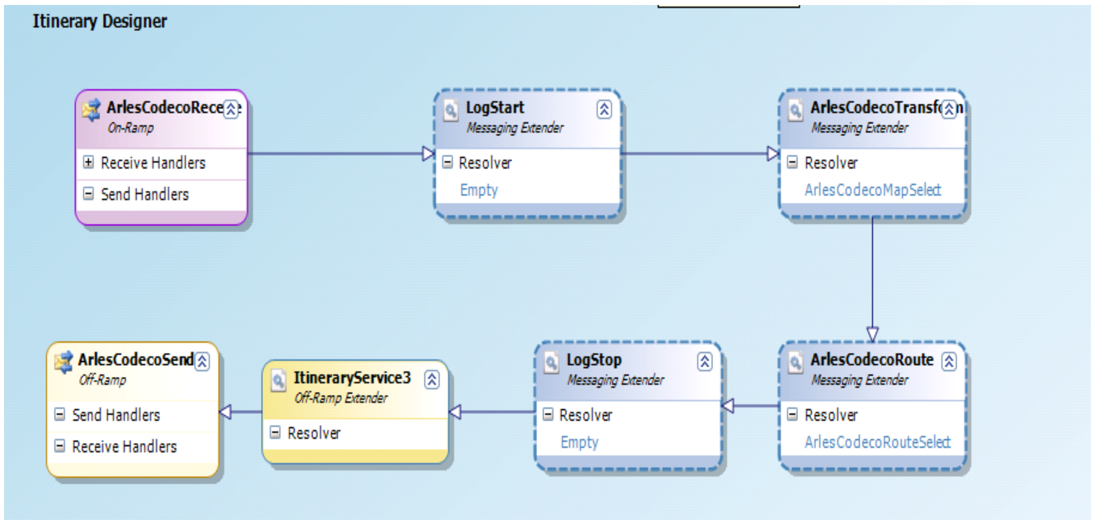


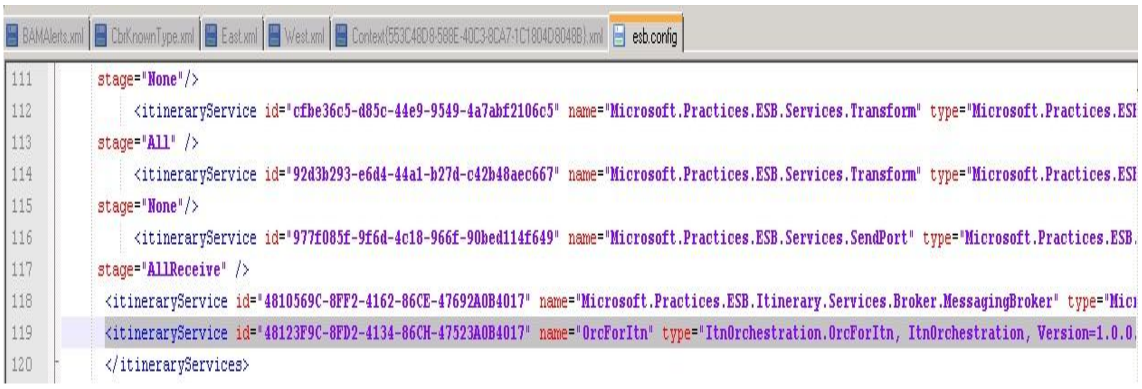
Figure 5.4. Itinerary design and service list

- Using Itinerary Services

  To implement a SOA based solution via ESB toolkit, it must be designed within itinerary to use which of services to be executed during message life cycle. ESB Toolkit supports two types of services: Messaging Service and Orchestration Services.

  - Messaging Service is a component that is called by pipeline that receives the message. The dispatcher component of calling pipeline processes the list of services that are message based and are attached to the incoming message. Two services are built in messaging service inside toolkit which is Transformation And Routing Service. These services can be increased by implementing new Message Based Services. I added three custom messaging services to adapt ESB Toolkit services to service based container logistics information framework. These three services are message based services, can be used in itinerary designer and processed in pipelines.

- Orchestrations are Business Process Management tools for BizTalk server and almost every integration needs these components to provide connectivity. However ESB Toolkit is a service oriented platform based on service consumption. It is not useful with implementing business logic through connectivity and integration. Microsoft gives the ability to add a custom orchestration to be used inside toolkit as a service. After orchestration is deployed to BizTalk Admin Console, its assembly information needs to be inserted to ESB.Config file as a new service. In this way, that custom orchestration can be seen and used in itinerary designer.



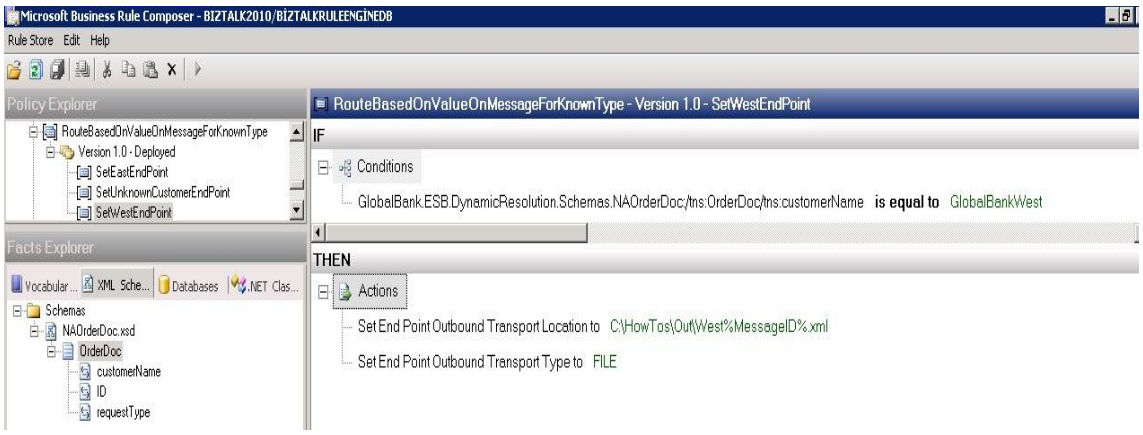
```
        stage="None"/>
            <itineraryService id="cfbe36c5-d85c-44e9-9549-4a7abf2106c5" name="Microsoft.Practices.ESB.Services.Transform" type="Microsoft.Practices.ESI
        stage="All" />
            <itineraryService id="92d3b293-e6d4-44a1-b27d-c42b48aec667" name="Microsoft.Practices.ESB.Services.Transform" type="Microsoft.Practices.ESI
        stage="None"/>
            <itineraryService id="977f085f-9f6d-4c18-966f-90bed114f649" name="Microsoft.Practices.ESB.Services.SendPort" type="Microsoft.Practices.ESB.
        stage="AllReceive" />
         <itineraryService id="4810569C-8FF2-4162-86CE-47692A0B4017" name="Microsoft.Practices.ESB.Itinerary.Services.Broker.MessagingBroker" type="Mic
         <itineraryService id="48123F9C-8FD2-4134-86CH-47523A0B4017" name="OrcForItn" type="ItnOrchestration.OrcForItn, ItnOrchestration, Version=1.0.0.
         </itineraryServices>
```

Figure 5.5. Service Definition in ESB.Config

- Schema Usage

Incoming message needs to be matched with a schema that is deployed to BizTalk Server to be processed inside ESB Toolkit. To combine more than one integration into a SOA based solution, it is a better way to use single schema. By using single schema we can provide a content based resolution with Resolver Framework. With single message, we can use its content to define which map to transform and which end point to be routed at runtime.

ESB toolkit provides main advantages of Service-Oriented Architecture. After converting BizTalk integration we hope to have these benefits in our solution. Thereby run-time configuration, loosely coupled integration pattern and reusable services have a significant place in service oriented environment that need to be take into consideration. To achieve these main goals, ESB toolkit provides framework and services. Their ability to support run-time configuration is provided by Resolver Service and Dynamic Ports. In itinerary designer and BizTalk components, Resolver framework is an agent to get configuration parameters.

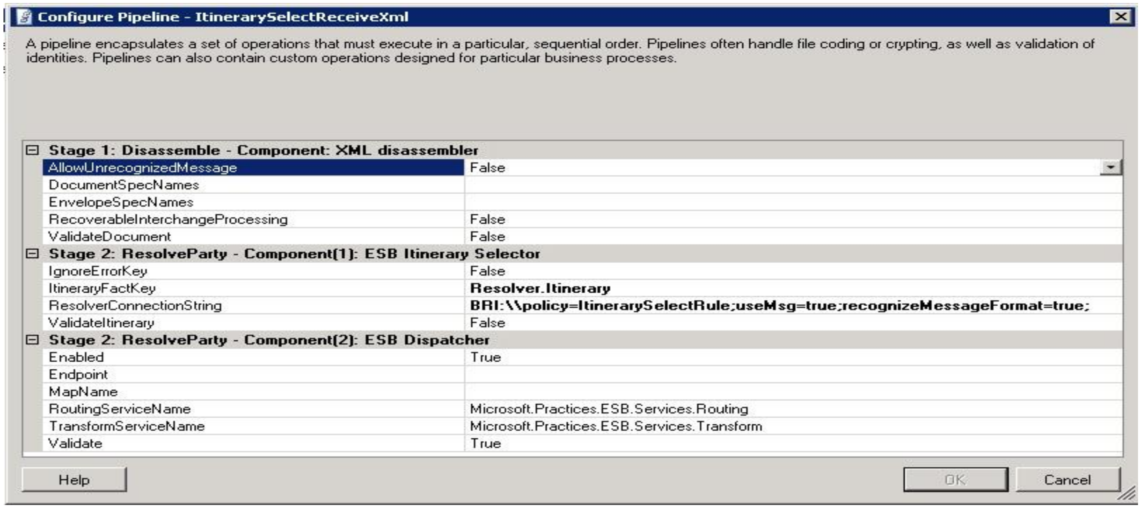

Microsoft Business Rule Composer - BIZTALK2010/BİZTALKRULEENGİNEDB
Rule Store Edit Help
Policy Explorer
RouteBasedOnValueOnMessageForKnownType
Version 1.0 - Deployed
SetEastEndPoint
SetUnknownCustomerEndPoint
SetWestEndPoint
Facts Explorer
Vocabular... XML Sche... Databases .NET Clas...
Schemas
NAOrderDoc.xsd
OrderDoc
customerName
ID
requestType

RouteBasedOnValueOnMessageForKnownType - Version 1.0 - SetWestEndPoint

```
IF
  Conditions
    GlobalBank.ESB.DynamicResolution.Schemas.NAOrderDoc:/tns:OrderDoc/tns:customerName  is equal to  GlobalBankWest
THEN
  Actions
    Set End Point Outbound Transport Location to  C:\HowTos\Out\West%MessageID%.xml
    Set End Point Outbound Transport Type to  FILE
```

Figure 5.6. Dynamic configuration in Business Rule Engine

The first thing that can be implemented is a dynamic itinerary resolution. The best practice for itineraries is, for every type of a message, there must be a single itinerary. In our work ‘Arles Codeco Solution’ we have a single itinerary but we did not use dynamic itinerary selection. It is possible to use it with Resolver Framework by selecting itinerary according to incoming message schema type or schema context.



A pipeline encapsulates a set of operations that must execute in a particular, sequential order. Pipelines often handle file coding or crypting, as well as validation of identities. Pipelines can also contain custom operations designed for particular business processes.

Stage 1: Disassemble - Component: XML disassembler	
AllowUnrecognizedMessage	False
DocumentSpecNames	
EnvelopeSpecNames	
RecoverableInterchangeProcessing	False
ValidateDocument	False
Stage 2: ResolveParty - Component(1): ESB Itinerary Selector	
IgnoreErrorKey	False
ItineraryFactKey	**Resolver.Itinerary**
ResolverConnectionString	**BRI:\\policy=ItinerarySelectRule;useMsg=true;recognizeMessageFormat=true;**
ValidateItinerary	False
Stage 2: ResolveParty - Component(2): ESB Dispatcher	
Enabled	True
Endpoint	
MapName	
RoutingServiceName	Microsoft.Practices.ESB.Services.Routing
TransformServiceName	Microsoft.Practices.ESB.Services.Transform
Validate	True

Figure 5.7. Dynamic Resolution via Resolver Framework

The second one is using Resolver Framework for Transformation and Routing Services. These services are reusable components of ESB Toolkit. For combining three integration into one ESB Solution, Transform Service with Resolver Service is used to decide which map to execute. Content of the schema consists of an id and according to that id; transform service gets a map name from ResolverFramework. By this way, Transformation service provided reusability and Resolver framework gives chance to have agility to urgent changes. And also, adding new services to itinerary or new maps to be used provides flexibility to our solution. So our solutions for both BizTalk and ESB Toolkit can be structured in Figure 5.8 below,

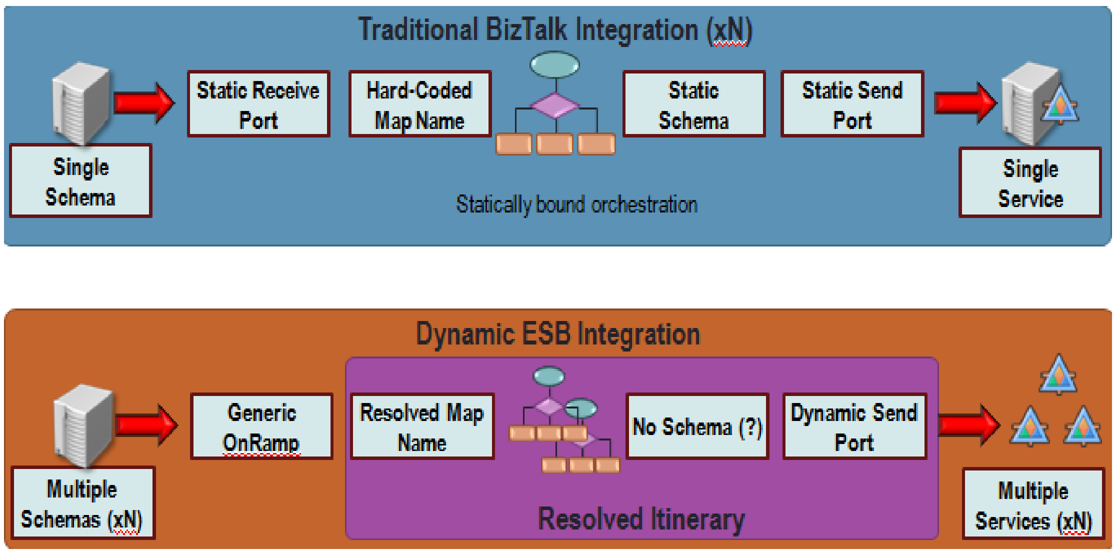


Figure 5.8. Classic vs. Proposed Integration Solution

CHAPTER 6

CASE STUDY in BİMAR

In this chapter It will be tried to explain the studies carried out in Bimar which performed the container logistics business integrations of the studies in real sector which were planned and touched upon in earlier chapters.

By adapting the present integrations onto the ESB Toolkit infrastructure, it was aimed to get benefit from the advantages of the SOA architecture which ESB applies and supports.

Integrations carried out within ESB Toolkit enable integration management with flexible, loosely-coupled and reusable services. It aims to circumvent repetitions of development, building and deployment processes which are called "hard-coded" as in classical BizTalk integrations and where all configurations are defined as static, and in which developers have to deal with during a change.

Within the company there are approximately 200 integrations in operation. After the domain and data analyses of these integrations, transferring the grouped integration clusters into ESB architecture as designated in previous chapter, new services created during these improvements and BizTalk ESB Toolkit components will be explained. Another crucial issue for the company and the real sector is the need for monitoring and exception which emerge after integration and process become more complex. This was implemented in production servers along with updated and improved integrations and was prepared to be put into production. Designed in accordance with ESB and SOA infrastructures, exception management and monitoring services work in harmony with the definitions of reusability, agility and flexibility which are the three basic features aforementioned in my thesis.

In the studies conducted, all relevant features of ESB Toolkit are utilized, integrations are applied as both orchestration-based and message-based. By activating the exception handling and portal usage, a new feature that is not provided by the present BizTalk architecture is obtained. Furthermore, resolvers providing dynamism and adapter infrastructures, along with exception capturing mechanisms and management portal, enable a rich platform. In figure below, it is seen that which components are used and developed in ESB Toolkit infrastructure. In figure 6.1 highlighted components of framework has been reused, developed and improved in custom framework services. Custom components for BizTalk Receive and Send ports are included in framework. Generated ESB based integration solutions consumes s ESB Toolkit core services, itinerary services also uses business rule engine, transformation engine and resolver-adaptor framework.

The studies and developments conducted within the domain of Enterprise Service Bus architecture will be elaborated on under the chapters below.

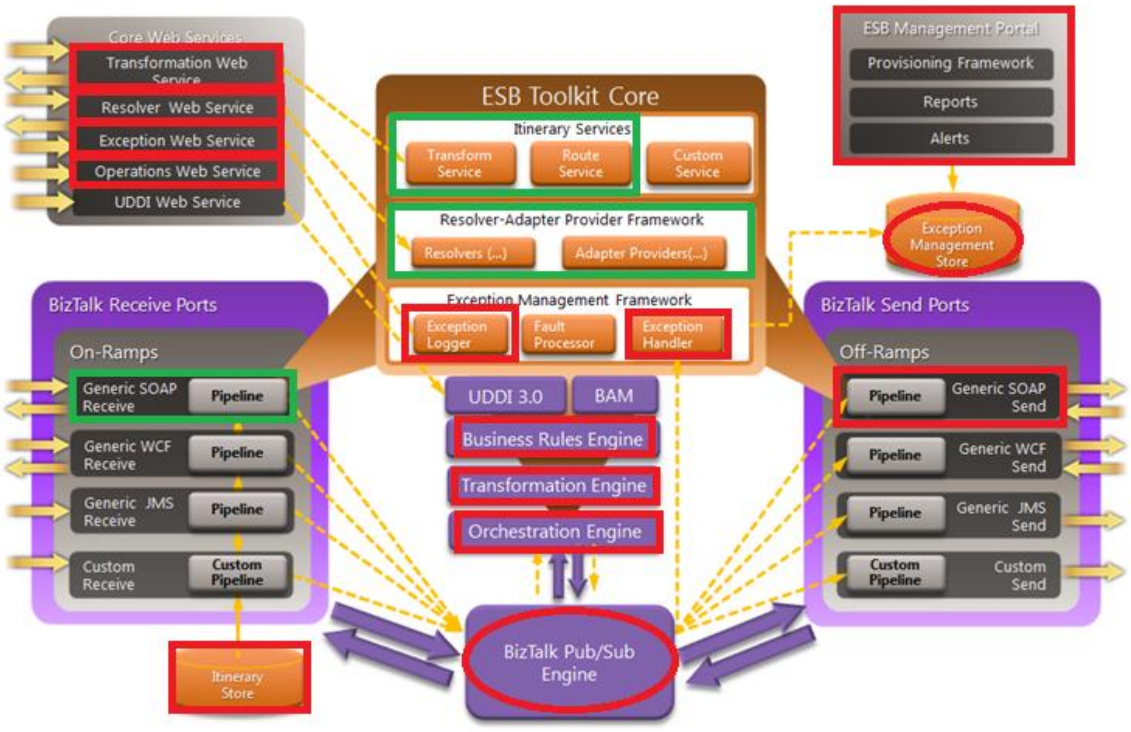


Figure 6.1. ESB Toolkit infrastructure with custom framework [40]

Within the scope of framework studies, for integrations that will be implemented in ESB architecture, development of Custom Services, how services should be developed and the purpose they serve for are explained. Framework software providing SOA architecture features, running container logistics business processes and enabling the development of new integrations in the face of domain needs with minimum effort has been developed.

In Integration Studies sub-chapter, improvements to run integrations with newly-developed ESB framework and integration details are further explained.

In Integration Management sub-chapter, however, how created framework services and running integrations work and controlled, the developments directed towards the end users are explained.

6.1. Service Oriented Integration Framework with ESB Toolkit

For integrations to run in ESB toolkit architecture, the need for interoperability, exception management, logging, integration status checking led to the need for developments on infrastructural basis. Since Exception Management and Portal infrastructures, features of ESB Toolkit, do not fully meet the needs, a brand new service and infrastructure software which would integrate with the exception management and the integration monitoring databases within Bimar was developed in accordance with ESB infrastructure.

The infrastructure used in classic integration management was realized in ESB platform as well. As this framework uses BizTalk Integration principles as running logic, integrations have been made to adapt it ESB Toolkit infrastructure. The details of this implementation are below.

Since Enterprise Service Bus architecture is a message-based architecture, message-based services have been developed. The one utilized in Microsoft platform, however, technically enables development of both message-based and orchestration-based services. Nevertheless, for Microsoft, orchestration tool was developed as message-based services since it was designed for the management and development of business logic, promising low quality service in terms of reusability.

Within ESB, it is possible to transform messages with a custom messaging service and to check whether it is a valid or correct message, or to enable handling of a set of processes suitable for the business flow.

To develop a message-based custom service in ESB Toolkit architecture, it is necessary to define a class applying an interface called ImessagingService. Via this class interface, 2 methods are implemented and 2 features are added to the class. To explain these methods briefly:

Execute: the part where which processes the service is responsible for carrying out within the method is coded. Within this method, the message is processed and the processed message is sent back, which is the most crucial part of the written service.

ShouldAdvanceStep : The method which designates whether a processed message will be processed by the next service in the itinerary list. Boolean return type true, however, if the return Boolean value is true, calling pipeline component put the next services on the itinerary into execution and message continues to be processed. If the service returns false on this method, service does not operate the next service. Some properties to be briefly touched upon in this class are:

Name : Service name seen on the itinerary. When the created service is defined in ESB.config file, it needs to share the same name with the property defined within the class.

SupportDissassemble : It designates whether the service written with this property support disassemble property or whether it enables multiple resolvers to operate or not.

Custom services are developed with the aim of exception management and logging. Services, as aforementioned, operate as message-based. All services can be called upon on the itinerary and the itinerary can be used during designing. Performed services can be categorized into 3 sub-groups: Exception handling transformation service, services opening and closing registry for logging. In figure 6.2 it is seen that, custom services are used in an itinerary. These services are executed in ESB while message is processed. Besides, custom services are built as reusable components, which any other itineraries can include. Each message inside framework can be logged, transformed and any occurring exceptions during message processing are kept in exception database.

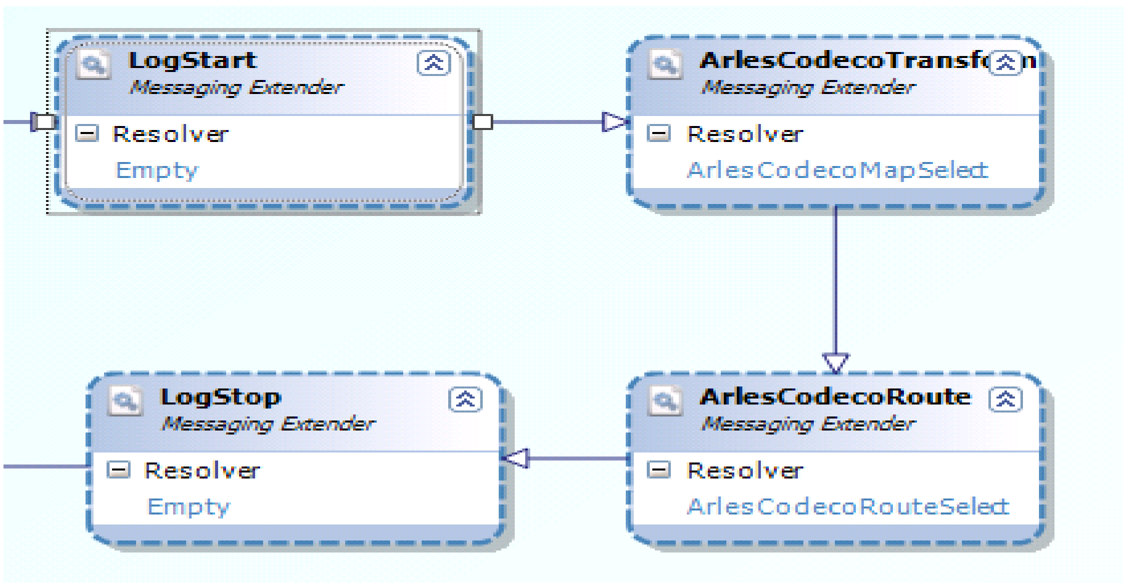


Figure 6.2. SOA based ESB Services

Log Registry Service: Through Log Registry Service, before a message received into an integration running on ESB toolkit is processed, a new registry is created for it to be logged in the framework. The registry here is started according to the GUID data of the itinerary which belongs to the message. Having been under registry, message continues to be processed by the following services. While Log Registry Service is dependent upon orchestration objects working for the message in the infrastructure which is developed for classic BizTalk integration, log service developed for ESB is totally different from this operating mechanism.

Operating message-based, ESB toolkit deals with services one by one with the help of dispatcher tool on BizTalk pipeline. This makes it impossible for orchestration objects to emerge in BizTalk infrastructure, thus it is necessary to use ESB toolkit itinerary features to save the messages. These data are captured during message flow and saved in the database in the infrastructure via log registry service.

Exception Handling Transform Service: Classic BizTalk integrations within Bimar encounter many exceptions during message transforms. During this process, exception handling and registry, continuation of the process are crucial for integrations. With this custom service, Transform Service within ESB has been developed to handle exceptions and these exceptions are kept in registry in accordance with the message and its itinerary GUID.

Transform used by ESB Toolkit service works similar to a service. The difference is, during the transform process, if the scripts used in the called map encounter an exception, exception objects in the script are firstly handled by core transform service of the ESB Toolkit, then by the transform exception handling service of the framework we developed. In this way, a healthy processing continuum will be sustained for the map called in transform service, message will not suffer from an exception, and occurring exceptions will be saved in control. After the exceptions are saved, messages can be viewed and exception details are accessible.

Log Close service: With this service, messages passing through intermediary services without any exceptions are registered as “successful” as they pass through the integration process without any glitch. This service is usually the last service of the itinerary file created for a message. If the previous services operated and the last operating service is Log Close service, this means that other services work without an exception and the message of Log Close service at that time is added to the system as “successful”. If, before this service, there may be a glitch in services in itinerary e.g.

Transform Exception Handling, that service will automatically handle the exception and add the message registry to the system as “Exception”. Services sustain integration management by operating on ESB toolkit infrastructure and services.

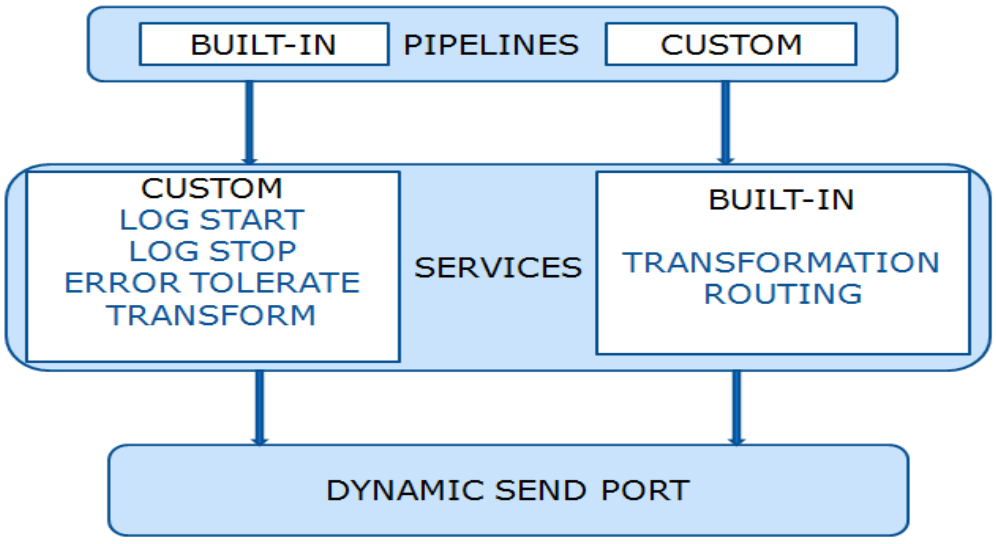


Figure 6.3. SOA based ESB Toolkit Framework

Seen from the figure 6.3, SOA Based ESB Toolkit Framework provides custom components for container logistics information systems. Custom pipelines enables to process Electronic Data Interchange For Administration, Commerce and Transport files (EDIFACT). Exception handling, integration tracking and monitoring features is supplied by log start and log stop services. Input file is processed by custom pipelines and executed in custom ESB services.

6.2. Service Oriented Integration of Logistic Processes

For the integrations realized within Bimar, framework support is utilized. These integrations work between different companies and systems located in logistical domain. In the table below, you can see on which domain the integrations run, integration code which takes a unique value and provides follow-ups of the integration within Bimar and the target/source systems of the integrations. Last column demonstrates which integration works with which ESB type in the newly-developed ESB architecture. Here is the list of the integrations:

Table 6.1. Integrations to work with custom framework

Integration type	**Integration Code**	**Source System**	**Target System**	**ESB Type**
Depot	ENT0000177	EDS	YNA	Orchestration
Seaway-Depot	ENT0000187	YNA	EDS	Orchestration
Port - Seaway	ENT0000171	Yılport	YNA	Orchestration
Port - Seaway	ENT0000171	Yılport	YNA	Message-based
Port - Seaway	ENT0000040	Navis	YNA	Orchestration
Port - Seaway	ENT0000040	Navis	YNA	Message-based
Roadway	ENT0000014	YNA	Catlogic	Orchestration
Port - Agency	ENT0000067	Arles	BSA	Message-based
Port - Agency	ENT0000165	Arles	HAPAG	Message-based
Port - Agency	ENT0000208	Arles	MSC	Message-based
Port - Agency	ENT0000050	Arles	Navis	Message-based
Port - Agency	ENT0000173	Arles	Marport	Message-based
Port - Agency	ENT0000056	Arles	YNA	Message-based

These integrations and the present ones have been grouped and started to run as a single integration. In this way multiple integrations work as single integration. Incoming and Outgoing point in ESB infrastructure have been reduced to single point. Integrations ENT0000165, ENT0000050 and ENT0000208 are good examples for this. Three different integrations running here have been grouped according to their data and process similarity and re-factored to run as a single integration, thus reducing the incoming and outgoing channel to a single one, and made to co-run thanks to the framework.

Merged integrations have been changed as Message-based and optimized in a way to enable multiple processes through a single integration. The use of Business Rule Engine Tool made dynamic end-point resolving possible. With the help of business rules which enable 3 different integrations to work as single ESB integration, and differentiate present integration processes, processes have become manageable. Being independent of code and integration design, business rules ensured ESB architecture to access these rules in run-time and run them, hence enabling application of the agility concept. For instance, whereas a change happening in the end-point that a file should reach requires a change in the end point definition which is bound as static in classic BizTalk integration process, and then taking it on build, deploy and production. A change by means of business rules, however, is just saved, run in run-time by ESB and is immediately taken into production. At the same time, the processes of taking it on build deploy and production server is not needed.

Container logistics process integrations merged and adapted to ESB framework with custom services. In Table 6.1 full list of integrations are seen that is worked on custom framework. Total number of integrations 13, merged number of integrations is 8. In Table 6.2 merged integrations are listed, and Table 6.3 is the list of integrations that is adapted to custom framework without merging with other integrations.

Table 6.2. List of merged integrations

Integration type	**Integration Code**	**Source System**	**Target System**
Port - Seaway	ENT0000171	Yılport	YNA
Port - Seaway	ENT0000040	Navis	YNA
Port - Agency	ENT0000067	Arles	BSA
Port - Agency	ENT0000165	Arles	HAPAG
Port - Agency	ENT0000208	Arles	MSC
Port - Agency	ENT0000050	Arles	Navis
Port - Agency	ENT0000173	Arles	Marport
Port - Agency	ENT0000056	Arles	YNA

Integrations that are not merged are included to system that uses custom services of framework. They behave as a ESB integration, use custom log and transformation services, support dynamic routing and runtime configuration but lack of similarity about input and output files or business logic, prevent them to be merged with other integrations.

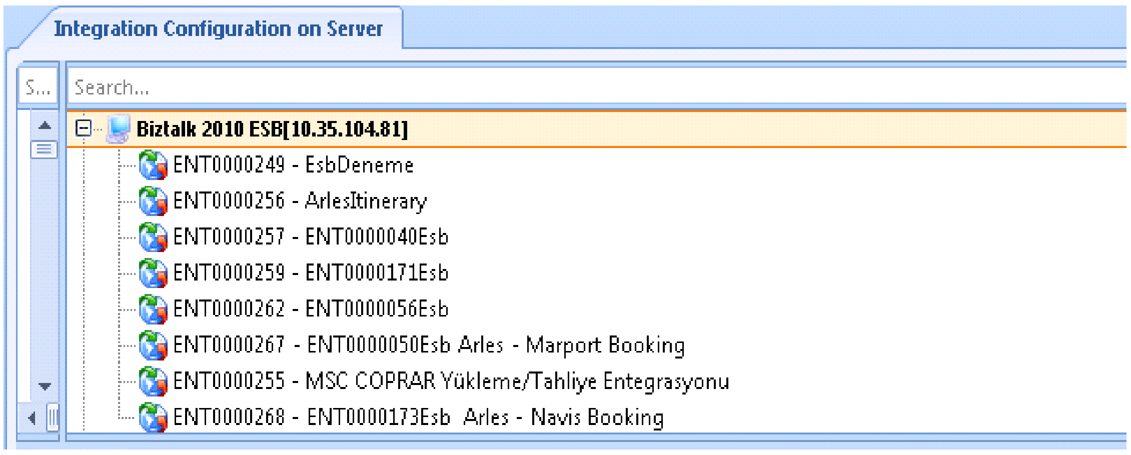
6.3. Integration Management in SOA based ESB Framework

Owing to services developed for ESB toolkit and the used framework, a healthy infrastructure for the integration management has been prepared and infrastructural improvements have been done for crucial problems like file follow-up and exception management.

Integrations run on ESB Toolkit have become easy to follow-up through a central system. Written custom services made it possible to monitor the integration process in the control mechanism.

ESB toolkit integrations have started to be logged on the basis of itinerary and registered as singular logs according to their itinerary GUID. A healthy start in integration, running transform service in a exception-tolerating fashion, and finding no exceptions during message processing enabled a successful registry log. On a contrary situation, message will be saved in the system as exception.

In the figure below, there is a list of integrations demonstrating that the running integrations on ESB server are under control of the developed infrastructure.

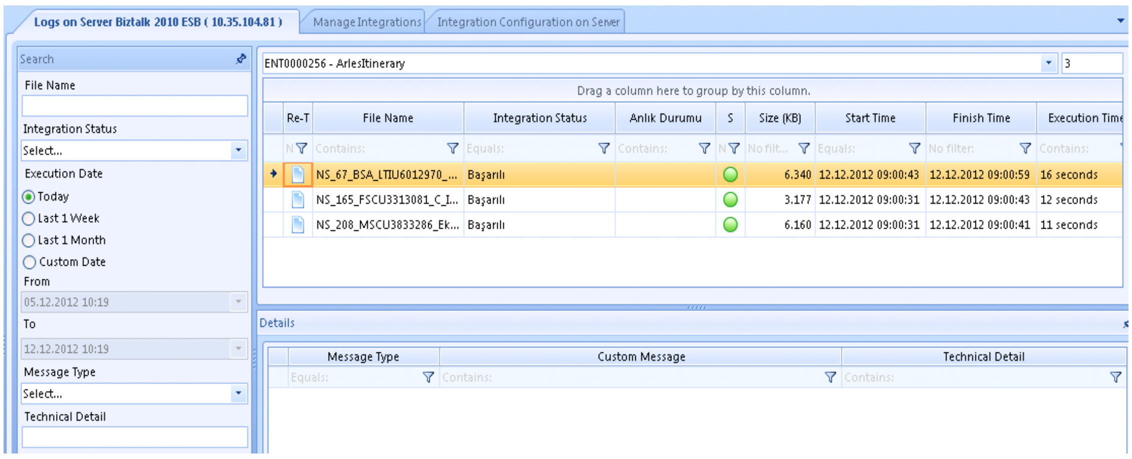


Biztalk 2010 ESB[10.35.104.81]
ENT0000249 - EsbDeneme
ENT0000256 - ArlesItinerary
ENT0000257 - ENT0000040Esb
ENT0000259 - ENT0000171Esb
ENT0000262 - ENT0000056Esb
ENT0000267 - ENT0000050Esb Arles - Marport Booking
ENT0000255 - MSC COPRAR Yükleme/Tahliye Entegrasyonu
ENT0000268 - ENT0000173Esb Arles - Navis Booking

Figure 6.4. Container logistic integration list

Each integration definition work with a itinerary file that belongs to itself and it is logged as single according to the itinerary GUID given to this itinerary in run-time. With the help of logging, it is possible to know if the messages coming to the integration processes are processed as “successful” or “fail” and to see the exception details of the failed integrations. The system, also, enables display of original files by recording all incoming messages or it may as well send a “fail” message to ESB again for processing.

Next image shows that integration management is enabled with the registry system provided by this infrastructure. With features like message follow-up, exception monitoring, it is possible to monitor ESB Toolkit integration processes.


Logs on Server Biztalk 2010 ESB (10.35.104.81) | Manage Integrations | Integration Configuration on Server

Search

File Name:

Integration Status: Select...

Execution Date: [✓ Today] Last 1 Week / Last 1 Month / Custom Date

From: 05.12.2012 10:19

To: 12.12.2012 10:19

Message Type: Select...

Technical Detail:

ENT0000256 - ArlesItinerary | 3

Drag a column here to group by this column.

Re-T	File Name	Integration Status	Anlık Durumu	S	Size (KB)	Start Time	Finish Time	Execution Time
N	Contains:	Equals:	Contains:	N	No filt...	Equals:	No filter:	Contains:
	NS_67_BSA_LTIU6012970_...	Başarılı		●	6.340	12.12.2012 09:00:43	12.12.2012 09:00:59	16 seconds
	NS_165_FSCU3313081_C_I...	Başarılı		●	3.177	12.12.2012 09:00:31	12.12.2012 09:00:43	12 seconds
	NS_208_MSCU3833286_Ek...	Başarılı		●	6.160	12.12.2012 09:00:31	12.12.2012 09:00:41	11 seconds

Details

Message Type	Custom Message	Technical Detail
Equals:	Contains:	Contains:

Figure 6.5. Integration and exception tracking

As seen in integration follow-up screen, ENT0000256 coded integration messages are run on itinerary file named ArlesItinerary. With all messages processed successfully, infrastructure developed to adapt the ESB makes it possible to observe technical data such as message size data, input and output time and processing time by the user and the developer. In the "Details" form below, it is possible to see the exceptions occurring in the messages.

Table 6.3. BizTalk vs. SOA based framework comparison

	Thesis Service Oriented Integration Infrastructure	**BizTalk Integration Infrastructure**
Reusable Services	YES	NO
Dynamic Routing	YES	NO
Exception Handling	YES	NO
Exception Monitoring	YES	NO
Integration Tracking	YES	PARTIALLY

Within the framework of the developed software, creating manageable, easy-to-follow-up, recordable integration of applications have been successful. The fact that the framework written on ESB toolkit is a widely-used structure is also a factor facilitating the realization of the targets.

CHAPTER 7

CONCLUSION

The focal point of thesis is service-based information systems for container logistics processes management. Thesis aims to tackle one of the most intricate business processes and corporate applications, that is, integration of applications and it tries to provide a solution infrastructure suitable to integration management. Study has been realized within Bimar Company which manages the business processes of Arkas in this domain.

First of all, after the analysis of business processes that are active in different domains by different companies, integration points based on this analysis were determined. A study was carried out to transfer the determined integrations to service-based architecture. With the advantages provided by the service-based architecture, a reusable, agile integration architecture that could optimize complicated business processes was obtained.

These studies were conducted on BizTalk Server, a product by Microsoft used at Bimar and on other Microsoft products. While existing integrations were on Hub and Spoke architecture that does not support integration optimization, they were transferred to ESB Toolkit by Microsoft and this enabled them to utilize the advantages of SOA architecture.

BizTalk Server was a good solution for a small number of integrations and for simple business processes, yet it did not yield the necessary performance in the complicated systems that work with different enterprises, like container logistics processes. SOA-based services defined in ESB Toolkit were, however, created with an agile infrastructure for quickly-changing business processes. Moreover, new services that could be added for the different enterprises with different technologies were provided with a flexible infrastructure. That the custom services could be used in all integrations without any configuration was an advantage brought by the reusability feature.

Moreover, with regards to exceptions and the monitoring of integrations, new services were written for an efficient process. For a healthy process, follow-up for integrations is as crucial as the monitoring and analysis of exceptions possible in business logic within integrations. In real life, those kinds of exceptions are possible to come across and to overcome this, the follow-up for exceptions are provided within ESB architecture.

To sum up, two of the recent needs and problems of business world, integration and management of business processes acquires an optimized structure in line with the targets determined in this thesis. The advantages provided by SOA and ESB architecture were utilized and a developable infrastructure was put into operation. With the architecture developed, exception and integration management was supported and the aimed service-oriented integration infrastructure for container logistics management was realized.

CHAPTER 8

FUTURE WORK

Container logistic integration framework handles exception monitoring, handling and integration tracking with custom ESB services. Reusable flexible and agile integration platform provides an optimum infrastructure for integration management. Business changes, existing integration merging and possible problems in container logistics can be maintained with less cost with custom ESB services. In addition to provided solution, an improvement can be done for maintaining business changes.

Agility term of SOA is provided with run time configuration inside ESB services. Resolver Adaptor Framework, Business Rule Engine usage is the key of supporting agility. Business changes and critical exceptions require minimum time cost and quick problem solution for a reliable communication. Therefore, usage of Business Rule Engine can be configured for also system users but also developers. In case of a critical problem or sudden change in business logic, system users can edit business rules and edited rules can be used in production environment without deployment process. This improvement increases system efficiency and remove bottleneck for integration management.

REFERENCES

[1] Samtani, G. and Sadhwani, D., EAI and web service: easier enterprise application integration? http://www.webservicesarchitect.com/content/articles/ samtani01print.asp

[2] Cummins, F. (2002), Enterprise Integration, New York: John Wiley.

[3] Sawhney, M. (2001), 'Don't Homogenize, Synchronize',Harvard Business Review, July-August 2001.

[4] DAVENPORT, T.H., 1993, Process Innovation: Reengineering Work through Information Technology,Harvard Business School Press,Boston, ABD.

[5] Brodie, M. and Stonebraker, M. (1995), Migrating LegacySystems, Morgan Kaufmann Publishers, San Francisco,CA.

[6] David S. Linthicum (1999), Enterprise Application Integration, Addison Wesley.

[7] Arsanjani, A.; Borges, B.; and Holley, K. Service-oriented architecture: Components and modeling can make the difference. Web Services Journal, 9, 1 (2004), 34–38.

[8] Newcomer, E., and Lomow, G. *Understanding SOA with Web Services.* Boston: AddisonWesley, 2004.

[9] Bell, (2008), M. Service-Oriented Modeling: Service Analysis, Design, and Architecture, Wiley

[10] Chappell, David (2004). Enterprise Service Bus. O'Reilly Media, Inc.

[11] http://www.techopedia.com/definition/5229/enterprise-service-bus-esb, last accessed on October 1st, 2013.

[12] http://www.microsoft.com/en-us/download/details.aspx?id=14293, last accessed on October 8th, 2013.

[13] http://www.microsoft.com/en-us/biztalk/what-is-biztalk.aspx, last accessed on October 8th, 2013.

[14] http://msdn.microsoft.com/en-us/biztalk/dd876606.aspx, last accessed on October 8th, 2013.

[15] http://msdn.microsoft.com/en-us/library/ff647958.aspx#intpatt-ch05_pointtopointconnection, last accessed on October 8th, 2013.

[16] http://www.poltman.com/en/technical-information/eai/topologies, last accessed on October 15th, 2013.

[17]
http://www.goldstonetech.com/investor%20info/white%20papers/EAI%20Overview.pdf, last accessed on October 15th, 2013.

[18]
http://ggatz.com/images/Enterprise_20Integration_20_20SOA_20vs_20EAI_20vs_20ESB.pdf, last accessed on October 15th, 2013.

[19] Tuğlular, T., Titiz Avcı, D., Çetin, Ş., Dağhan, G., Özemre, M., Oysal, T., "An Approach to Find Integration and Monitoring Points for Container Logistics Business Processes", 2012, The Fourth International Conferences on Advanced Service Computing, SERVICE COMPUTATION 2012, Nice, France

[20] B. Ganter and R. Wille, "Formal Concept Analysis: Mathematical Foundations", Springer, Berlin, 1999.

[21] A. Formica, "Concept similarity in Formal Concept Analysis: An information content approach", Knowledge-Based Systems, 21(1), pp. 80–87, 2008.

[22] Enterprise service bus allows IT to adapt to a fast-changing business environment. By: Huizen, Gordon Van, Computer Weekly, 00104787, 3/22/2005

[23] BizTalkESBToolkitDocs
http://www.microsoft.com/enus/download/details.aspx?id=11847, last accessed : 29.10.2013

[24] http://www.techopedia.com/definition/13984/logistics-management, last accessed on November 5st, 2013.

[25] Zhang, L., Li, J., Yu, M., "An Integration Research on Service-oriented Architecture (SQA) for Logistics Information System", 2006, IEEE International Conference on Service Operations & Logistics & Informatics; 2006, p1059-1063, 5p

[26] Fei, Z., Shufen, Liu., "Research and Application of the ESB Based on Agent in the Integration of the MIS in Power Plant", 2010, pp. 250-3. *Publisher:* Piscataway, NJ USA ; Beijing China: IEEE *Country of Publication:* USA

[27] Jan, Jiang and others, "Research on application of Web based ESB in School Common Data Platform ", 2009, 4th International Conference on Computer Science & Education

[28] Rajini, N., Bhuvaneswari, T. 2010 , International Journal on Computer Science and Engineering Vol. 2 Issue 6, p1980-1983

[29] Wu, J., Tao, X., "Research of Enterprise Application Integration Based-on ESB",2010, 2nd International Conference on Advanced Computer Control

[30] Seiringer, W., 2009, "Service-oriented Analysis of Logistics Services", Logistics and Industrial Informatics, 2009. LINDI 2009. 2nd International

[31] Yu, D., Yan, D., 2011, "Towards the Integration of Enterprise Service Bus with UDDI Server: A Case Study", International Conference on System Science and Engineering, Macau, China

[32] Hohpe, G., Woolf, B., (2003), Enterprise Integration Patterns, Addison Wesley

[33] http://www.scis.ulster.ac.uk/~zumao/teaching/COM720/readings/reading10.pdf last accessed on September 28th, 2013.

[34] http://integrella.com/what-is-soa/, last accessed on October 20th, 2013.

[35] http://en.wikipedia.org/wiki/Enterprise_service_bus, last accessed on October 20th, 2013.

[36] http://sandroaspbiztalkblog.wordpress.com/2011/11/01/article-microsoft-biztalk-server-seen-by-the-programmers-eyes/, last accessed on November 1st, 2013.

[37] http://www.comelio.com/en/business_solutions/integration/biztalkserver, last accessed on November 1st, 2013.

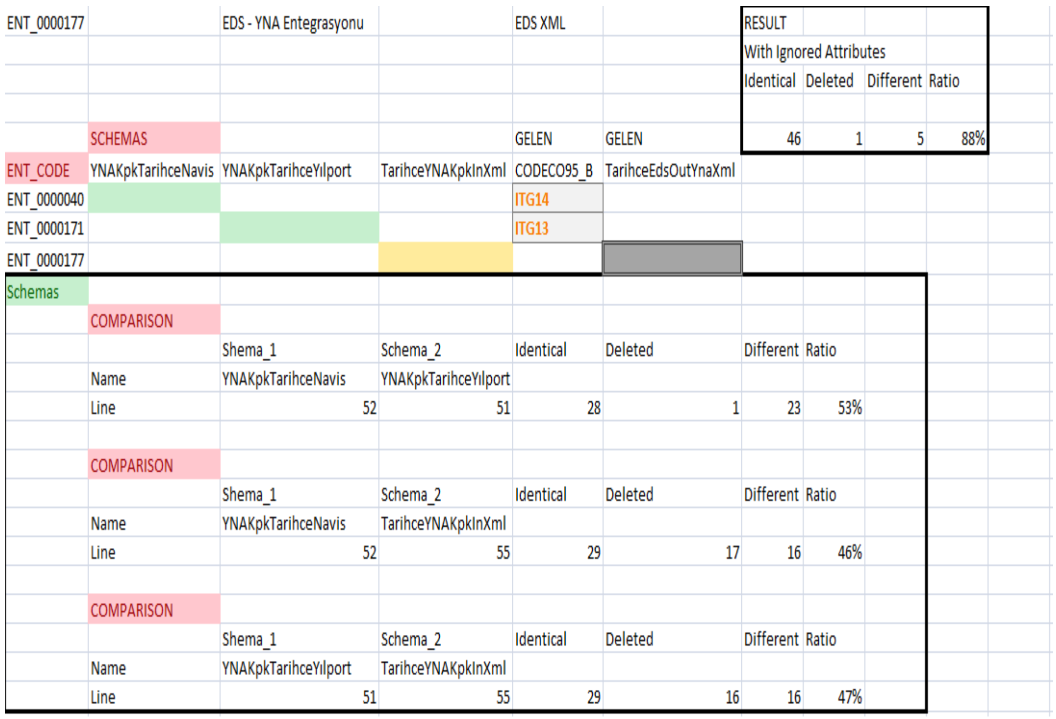
[38] http://msdn.microsoft.com/en-us/library/ff699598.aspx, last accessed on November 1st, 2013.

[39] http://msdn.microsoft.com/en-us/library/ee236726(v=bts.10).aspx, last accessed on November 1st, 2013.

[40] http://www.mikethearchitect.com/2009/06/microsoft-esb-toolkit.html%20, last accessed on November 1st, 2013.

APPENDIX A

BUSINESS PROCESSES ANALYSIS



ENT_0000177		EDS - YNA Entegrasyonu		EDS XML		RESULT			
						With Ignored Attributes			
						Identical	Deleted	Different	Ratio
	SCHEMAS			GELEN	GELEN	46	1	5	88%
ENT_CODE	YNAKpkTarihceNavis	YNAKpkTarihceYılport	TarihceYNAKpkInXml	CODECO95_B	TarihceEdsOutYnaXml				
ENT_0000040	🟩			ITG14					
ENT_0000171		🟩		ITG13					
ENT_0000177			🟨		▨				
Schemas									
	COMPARISON								
		Shema_1	Schema_2	Identical	Deleted	Different	Ratio		
	Name	YNAKpkTarihceNavis	YNAKpkTarihceYılport						
	Line	52	51	28	1	23	53%		
	COMPARISON								
		Shema_1	Schema_2	Identical	Deleted	Different	Ratio		
	Name	YNAKpkTarihceNavis	TarihceYNAKpkInXml						
	Line	52	55	29	17	16	46%		
	COMPARISON								
		Shema_1	Schema_2	Identical	Deleted	Different	Ratio		
	Name	YNAKpkTarihceYılport	TarihceYNAKpkInXml						
	Line	51	55	29	16	16	47%		

Figure A.1. Integration Data Similarity Comparison

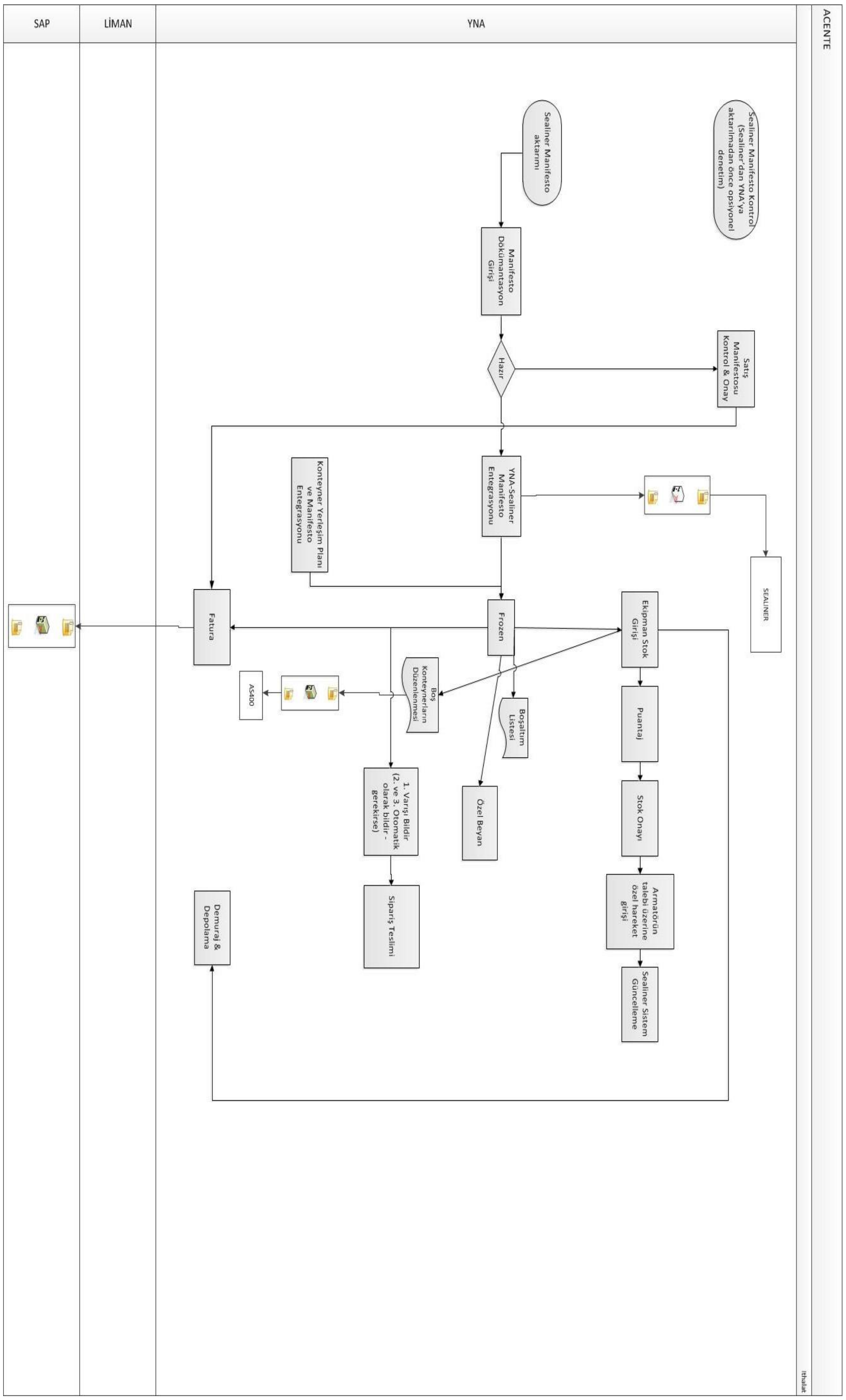

ACENTE
İthalat
SAP
LİMAN
YNA
Sealiner Manifesto Kontrol (Sealiner'dan YNA'ya aktarımadan önce opsiyonel denetim)
Sealiner Manifesto aktarımı
Manifesto Dokümantasyon Girişi
Hazır
Satış Manifestosu Kontrol & Onay
YNA-Sealiner Manifesto Entegrasyonu
Konteyner Yerleşim Planı ve Manifesto Entegrasyonu
SEALINER
Fatura
Frozen
Ekipman Stok Girişi
Boş Konteynerların Düzenlenmesi
AS400
Boşaltım Listesi
Puantaj
Özel Beyan
Stok Onayı
1. Varışı Bildir (2. ve 3. Otomatik olarak bildir - gerekirse)
Sipariş Teslimi
Armatörün talebi üzerine özel hareket girişi
Sealiner Sistem Güncelleme
Demuraj & Depolama

Figure A.2. Agency - Importation Business Process

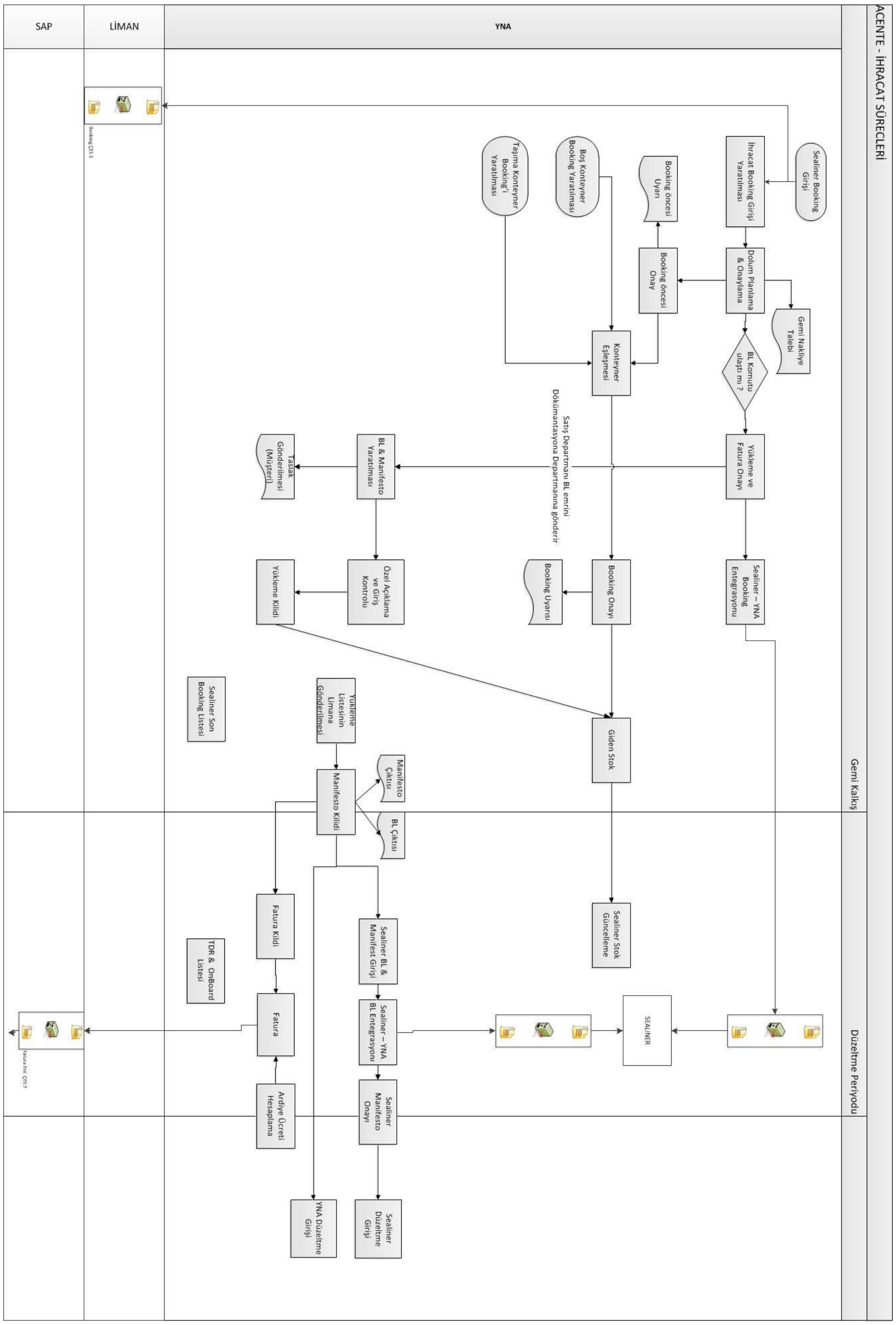


Figure A.3. Agency – Exportation Business Process

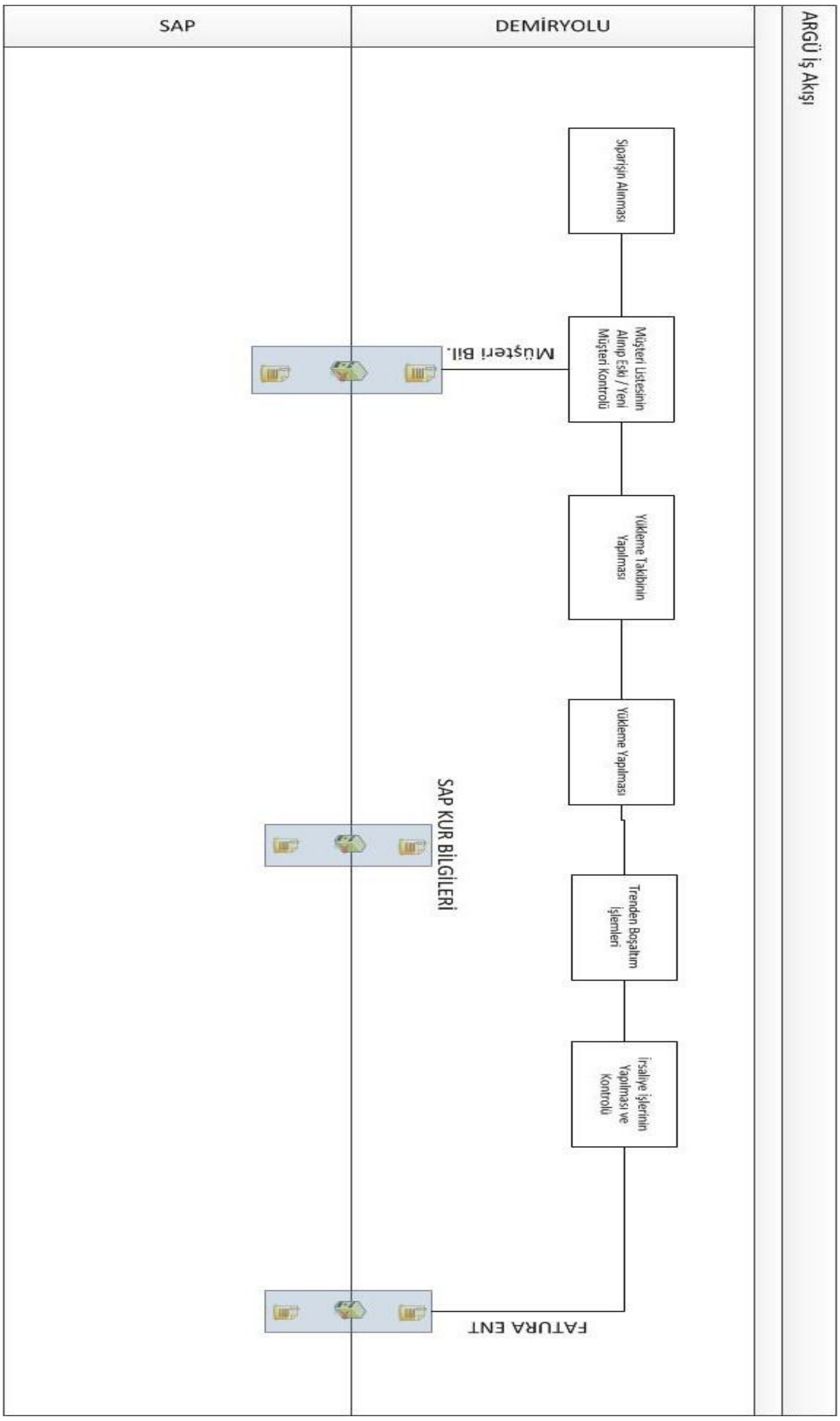

ARGÜ İş Akışı
DEMİRYOLU
SAP
Siparişin Alınması
Müşteri Listesinin Alınıp Eski / Yeni Müşteri Kontrolü
Müşteri Bil.
Yükleme Takibinin Yapılması
Yükleme Yapılması
SAP KUR BİLGİLERİ
Trenden Boşaltım İşlemleri
İrsaliye İşlerinin Yapılması ve Kontrolü
FATURA ENT

Figure A.4. Railway Business Process

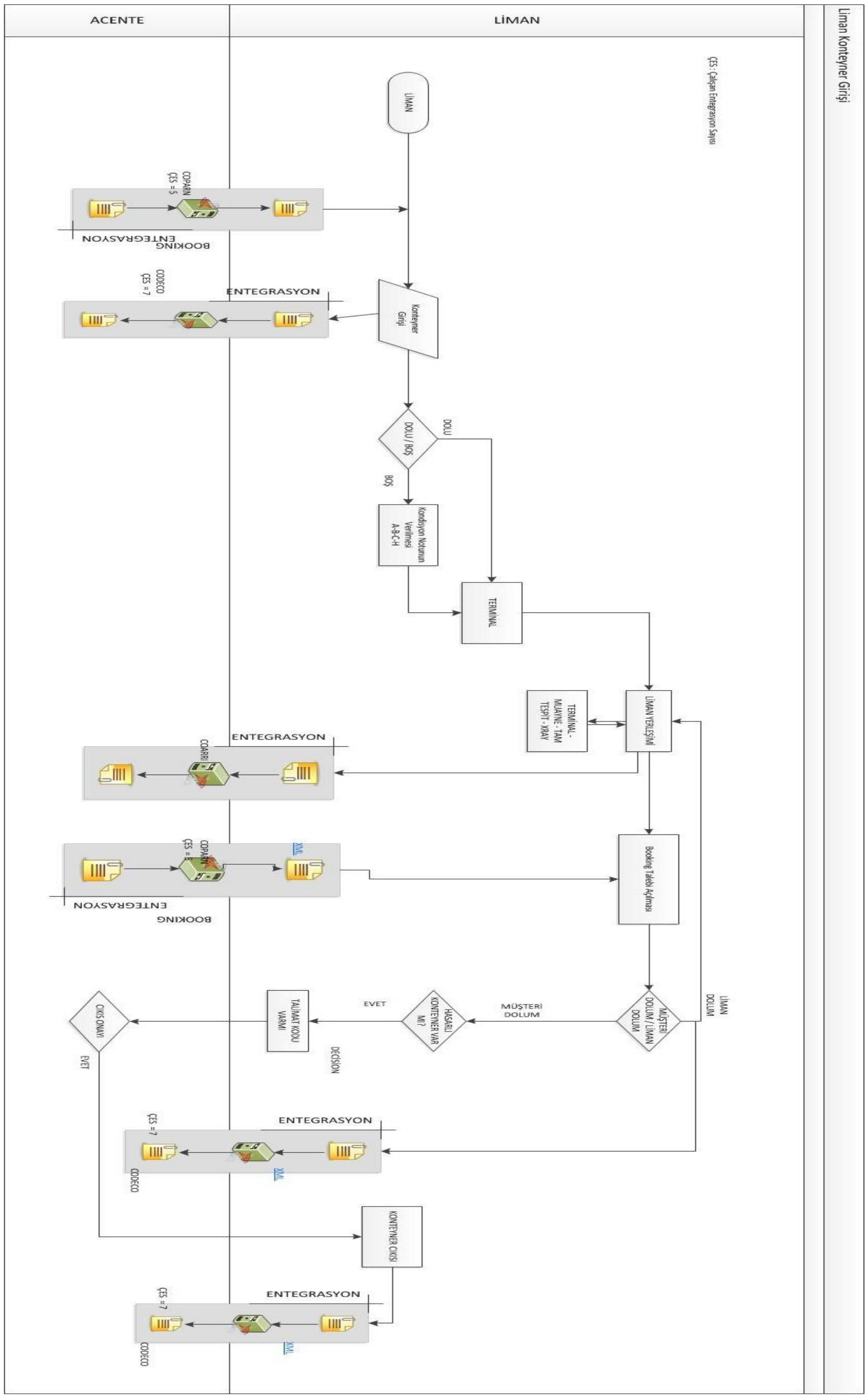


Figure A.5. Terminal – Container Entrance Business Process

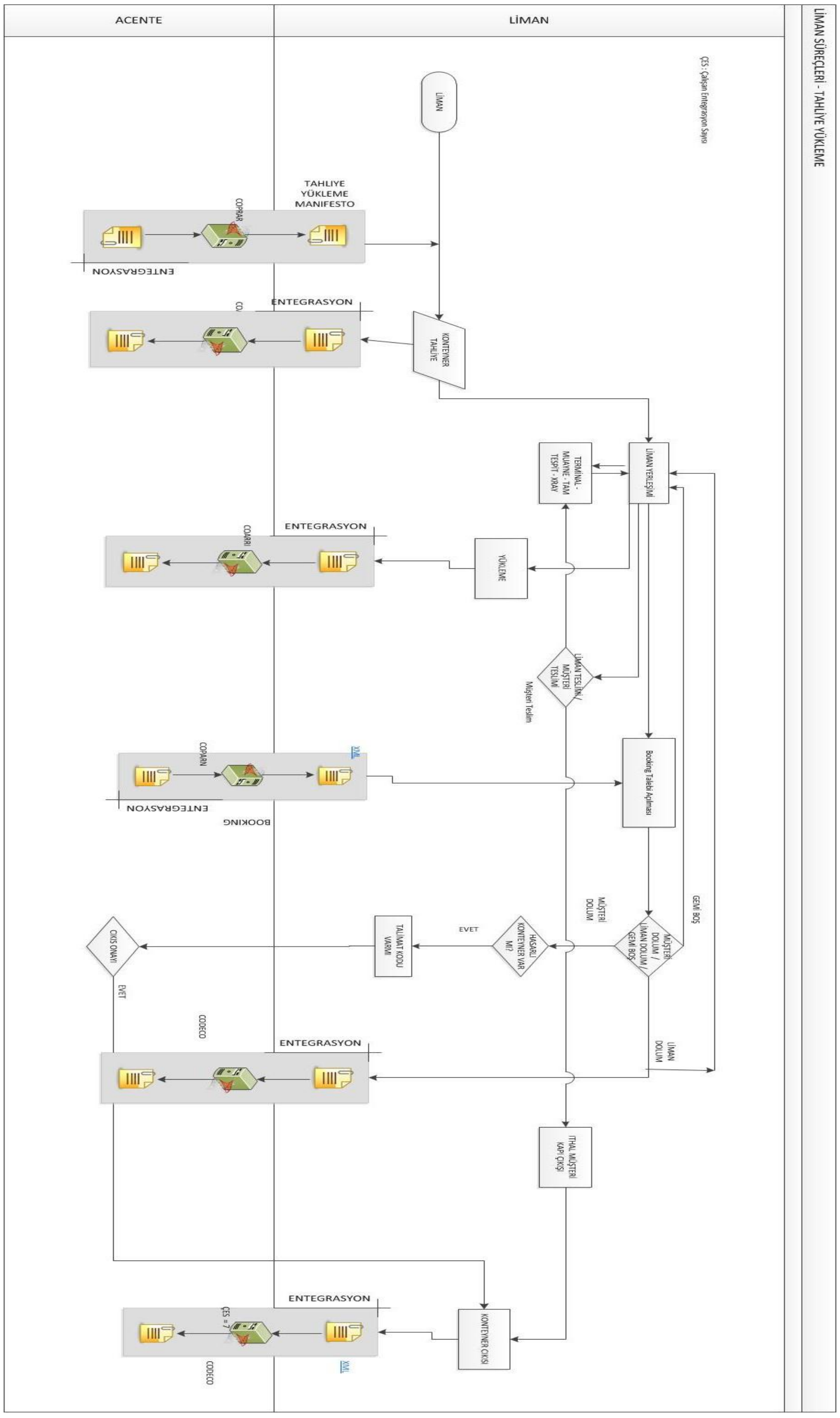

LİMAN SÜREÇLERİ - TAHLİYE YÜKLEME
ACENTE
LİMAN
ÇES : Çalışan Entegrasyon Sayısı
LİMAN
TAHLIYE YÜKLEME MANIFESTO
COPRAR
ENTEGRASYON
KONTEYNER TAHLİYE
LİMAN YERLEŞİMİ
TERMİNAL - MUAYENE - TAM TESPİT - XRAY
COARRI
YÜKLEME
LİMAN TESLİM / MÜŞTERİ TESLİM
Müşteri Teslim
COPARN
XML
BOOKING
Booking Talebi Açılması
MÜŞTERİ DOLUM / LİMAN DOLUM / GEMİ BOŞ
GEMİ BOŞ
MÜŞTERİ DOLUM
HASARLI KONTEYNER VAR MI?
EVET
TALİMAT KODU VAR MI
ÇIKIŞ ONAYI
CODECO
LİMAN DOLUM
İTHAL MÜŞTERİ KAPI ÇIKIŞI
KONTEYNER ÇIKIŞI
ÇES = 7

Figure A.6. Terminal – Loading/unloading Business Process

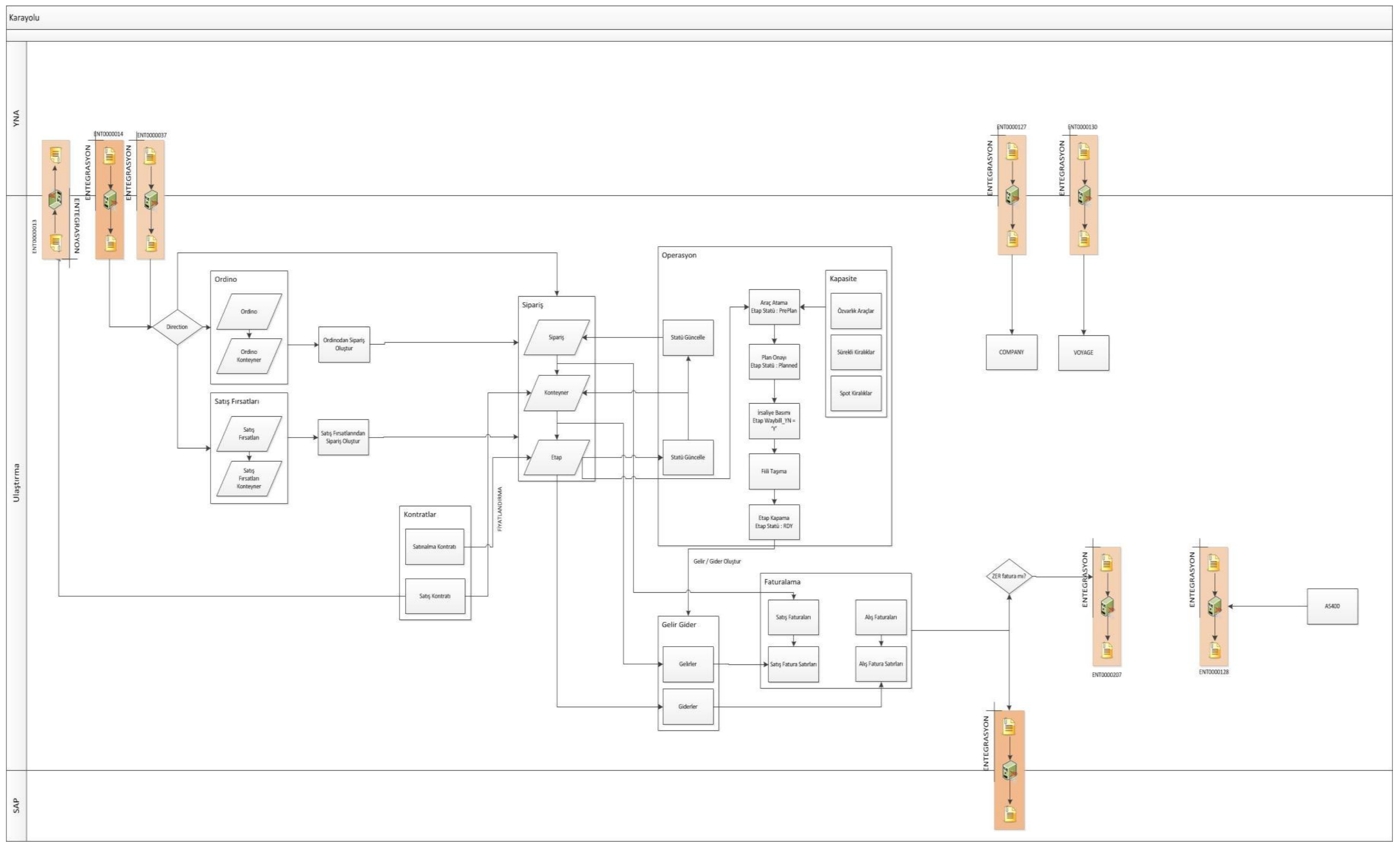


Figure A.7. Highway Business Process

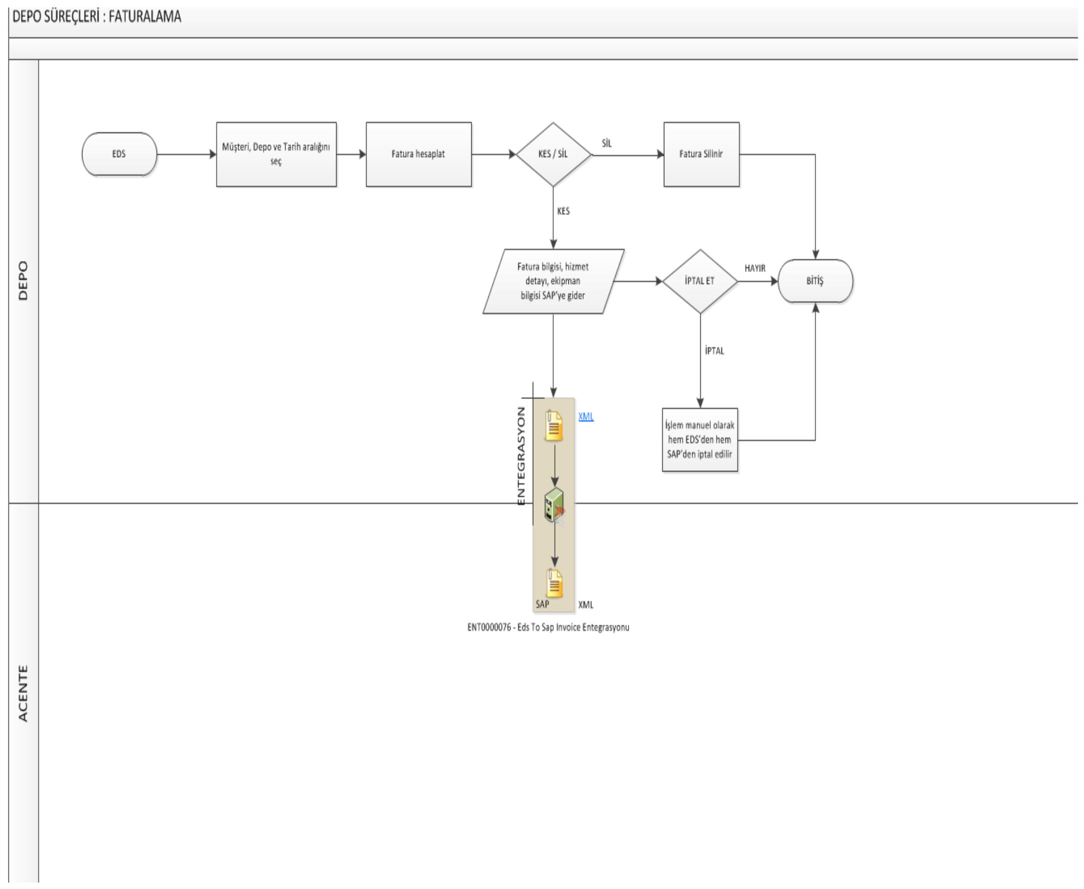

DEPO SÜREÇLERİ : FATURALAMA
DEPO
EDS
Müşteri, Depo ve Tarih aralığını seç
Fatura hesaplat
KES / SİL
SİL
Fatura Silinir
KES
Fatura bilgisi, hizmet detayı, ekipman bilgisi SAP'ye gider
İPTAL ET
HAYIR
BİTİŞ
İPTAL
İşlem manuel olarak hem EDS'den hem SAP'den iptal edilir
ENTEGRASYON
XML
SAP
XML
ENT0000076 - Eds To Sap Invoice Entegrasyonu
ACENTE

Figure A.8. Depot Invoice Process

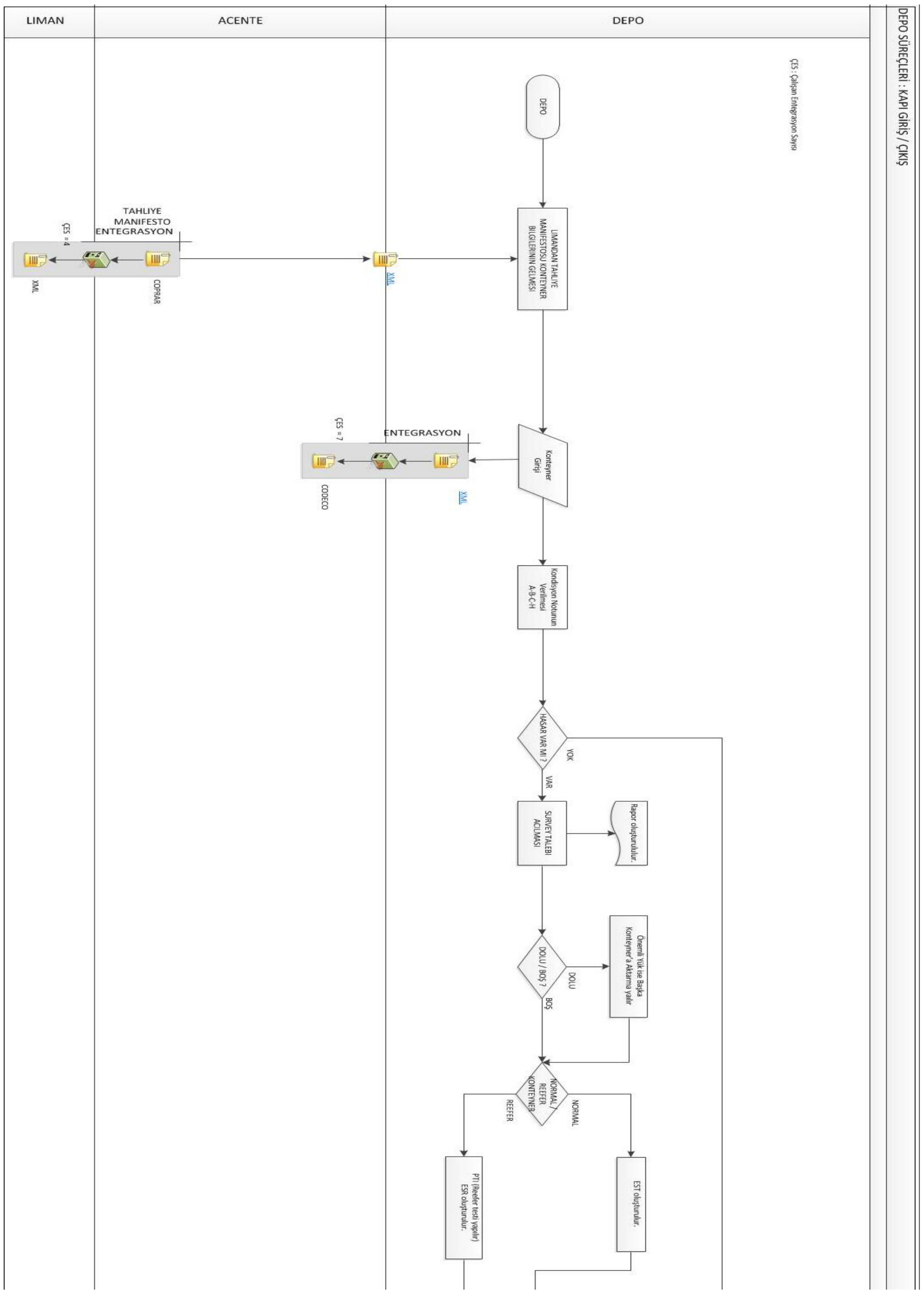

LIMAN
ACENTE
DEPO
DEPO SÜREÇLERİ : KAPI GİRİŞ / ÇIKIŞ
ÇES : Çalışan Entegrasyon Sayısı
DEPO
TAHLIYE MANIFESTO ENTEGRASYON
ÇES = 4
XML
COPRAR
XML
LIMANDAN TAHLIYE MANIFESTOSU KONTEYNER BILGILERININ GELMESI
ENTEGRASYON
ÇES = 7
CODECO
XML
Konteyner Girişi
Kondisyon Notunun Verilmesi A-B-C-H
HASAR VAR MI ?
YOK
VAR
SURVEY TALEBI ACILMASI
Rapor oluşturulur.
DOLU / BOŞ ?
DOLU
BOŞ
Önemli Yük ise Başka Konteyner'a Aktarma yapılır
NORMAL / REEFER KONTEYNER
NORMAL
REEFER
EST oluşturulur.
PTI (Reefer testi yapılır) ESR oluşturulur.

Figure A.9 Depot Business Process

(cont. on next page)

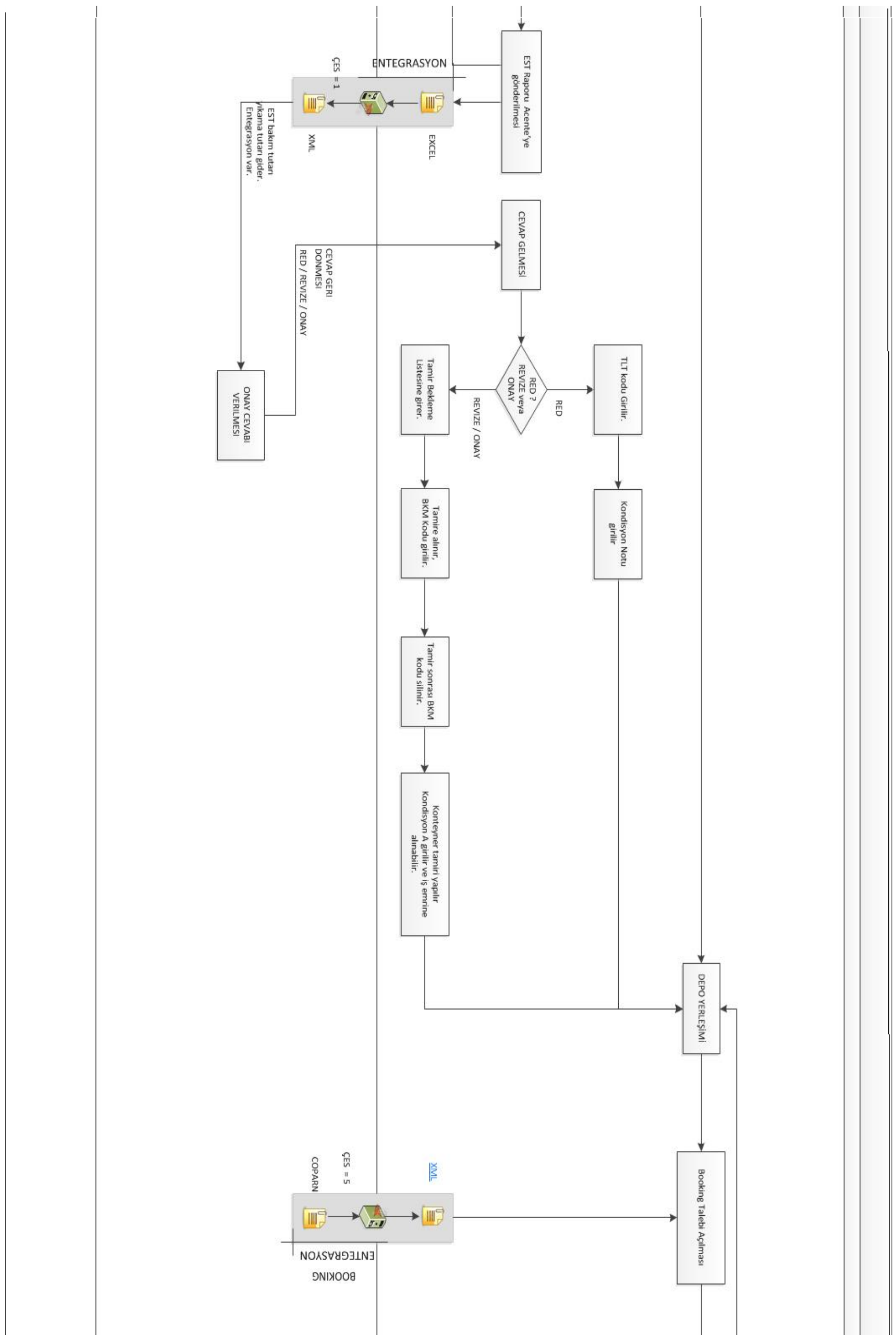

EST Raporu Acente'ye gönderilmesi
ENTEGRASYON
ÇES = 1
EXCEL
XML
EST bakım tutarı yıkama tutarı gider. Entegrasyon var.
CEVAP GELMESİ
CEVAP GERİ DÖNMESİ RED / REVİZE / ONAY
ONAY CEVABI VERİLMESİ
RED ? REVİZE veya ONAY
RED
REVİZE / ONAY
TLT kodu Girilir.
Kondisyon Notu girilir
Tamir Bekleme Listesine girer.
Tamire alınır, BKM Kodu girilir.
Tamir sonrası BKM kodu silinir.
Konteyner tamiri yapılır Kondisyon A girilir ve iş emrine alınabilir.
DEPO YERLEŞİMİ
Booking Talebi Açılması
XML
ÇES = 5
COPARN
ENTEGRASYON
BOOKING

Figure A.9 Depot Business Process (cont.)

(cont. on next page)

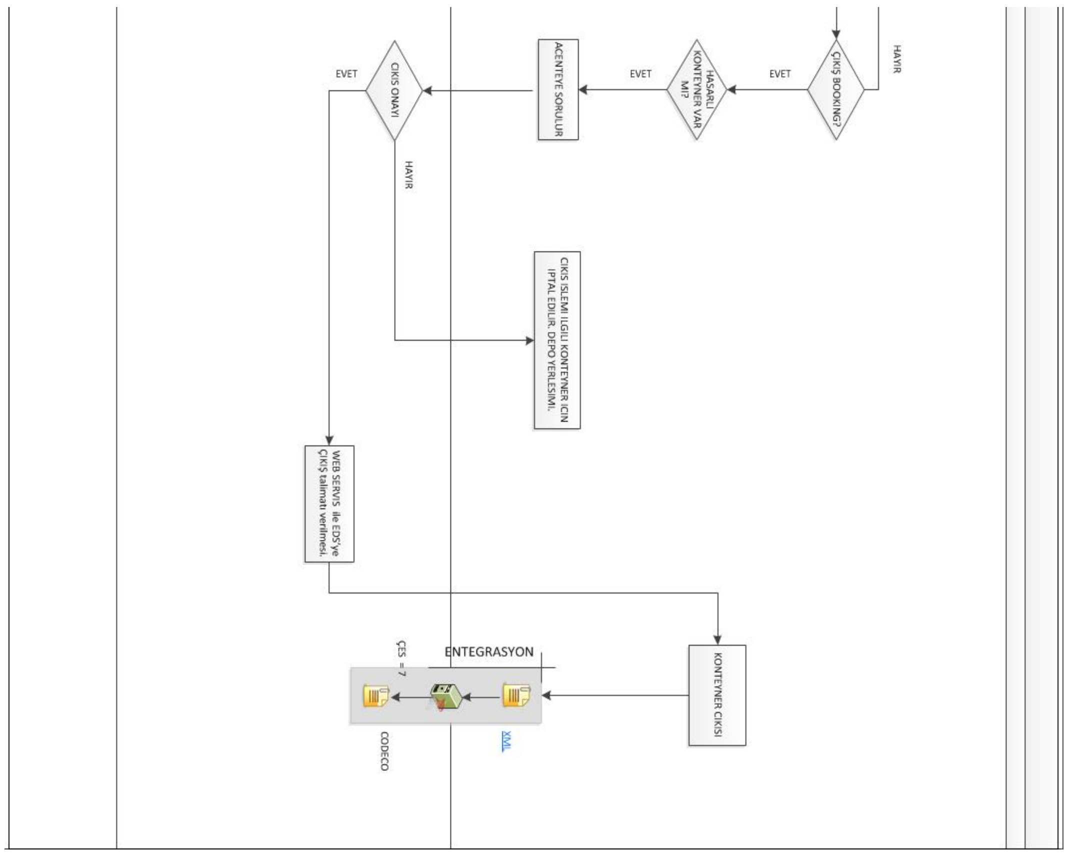

ÇIKIŞ BOOKING?
HAYIR
EVET
HASARLI KONTEYNER VAR MI?
EVET
ACENTEYE SORULUR
ÇIKIŞ ONAYI
EVET
HAYIR
ÇIKIŞ İŞLEMİ İLGİLİ KONTEYNER İÇİN İPTAL EDİLİR. DEPO YERLEŞİMİ.
WEB SERVIS ile EDS'ye ÇIKIŞ talimatı verilmesi.
KONTEYNER ÇIKIŞI
ENTEGRASYON
ÇES = 7
XML
CODECO

Figure A.9 (cont.)