TÉCNICO LISBOA

Service Integration Framework for TIBCO ActiveMatrix® BusinessWorks™ 6

Integration Framework

Diogo Rebelo da Cruz

Thesis to obtain the Master of Science Degree in

Information Systems and Computer Engineering

Supervisor: Prof. José Alberto Rodrigues Pereira Sardinha

Examination Committee

Chairperson: Prof. Francisco João Duarte Cordeiro Correia dos Santos
Supervisor: Prof. José Alberto Rodrigues Pereira Sardinha
Member of the Committee: Prof. Miguel Filipe Leitão Pardal

May 2018

Acknowledgments

I would like to thank my mother, sister and girlfriend for their friendship, encouragement and caring over all these years, for always being there for me through thick and thin and without whom this project would not be possible.

I would also like to acknowledge my dissertation supervisor Prof. José Alberto Rodrigues Pereira Sardinha for his insight, support and sharing of knowledge that has made this Thesis possible. Another important acknowledge is to Bruno Gouveia and Pedro Raposo, the persons who helped me every time i needed inside the company where this work was developed.

Last but not least, to all my friends and colleagues that helped me grow as a person and were always there for me during the good and bad times in my life. Thank you.

Finally, I would like to dedicate this thesis to my mother and sister for their friendship and caring during all my life, for always being there to support me and without whom this work would not be possible.

To each and every one of you – Thank you.

Abstract

The term Integration is a consequence of the creation and use of services. They have been around for some years now, but the way they are inserted in the real world, makes them very difficult to maintain and manage. Typically used as application integration mechanisms, these technologies require a specialized technical knowledge that is very hard to find.

In Portugal, only major clients in the telecommunication, banking and utilities areas, have the resources to implement application integration strategies in a considerable scale. One of the available product to implement these strategies is TIBCO ActiveMatrix® BusinessWorks™. Despite the fact that this is a very complete tool for integration, to develop well organized projects, it is also required to follow multiple guidelines, previously identified and established by several integration professionals. The use of these guidelines simplifies the work of all developers and allows the interchange of technical resources between them.

With the creation of the latest version of BusinessWorks™, these rules, architecture guidelines, development and deployment needed to be reevaluated, reviewed and implemented.

In this work, we proposed and developed a solution of a framework for the latest version of BusinessWorks™.

Keywords

Service-Oriented Architecture; Framework; Integration; Service; TIBCO; BusinessWorks; Process; Dissertation; Solution; Pilot; Architecture; Deployment; Development.

Resumo

As arquiteturas orientadas a serviços e micro-serviços já existem há alguns anos, no entanto a forma como são colocadas em prática no mundo real torna-as difíceis de gerir e manter. Usadas tipicamente como mecanismo de integração aplicacional, estas tecnologias de nicho requerem um conhecimento técnico especializado difícil de encontrar.

Em Portugal apenas grandes clientes na área das telecomunicações, banca e utilities têm recursos para implementar estratégias a larga escala de integração aplicacional, usando tipicamente produtos TIBCO(BusinessWorks™ 5) seguindo várias guidelines identificadas e estabelecidas pelos vários profissionais da área ao longo dos últimos 10 anos. A normalização que surge do uso destas guidelines permite criar sinergias entre as várias empresas que seguem estas boas práticas, e permite que recursos técnicos sejam trocados entre elas com curvas de aprendizagem muito reduzidas.

Com o surgimento da nova versão de TIBCO BusinessWorks™, estas regras e guidelines de arquitetura, desenvolvimento e deployment necessitavam de ser reavaliadas, revistas e novamente implementadas.

Neste documento encontra-se apresentada a solução desenvolvida pelo autor.

Palavras Chave

Arquitetura Orientada a Serviços; Integração; Serviços; TIBCO; Processo; Dissertação; Solução; Arquitetura; Desenvolvimento.

Contents

List of Figures

List of Tables

Acronyms

TIBCO	The Information Bus Company
BW	BusinessWorks
SFW	Service Framework
EI	Enterprise Integration
EAI	Enterprise Application Integration
IS	Information Systems
IT	Information Technology
OS	Operating System
URL	Uniform Resource Locator
UI	User Interface
XML	Extensible Markup Language
IST	Instituto Superior Tecnico
SOA	Service Oriented Architecture
ESB	Enterprise Service Bus
BP	Business Process
BPMN	Business Process Model and Notation
SOAP	Simple Object Access Protocol
WSDL	Web Service Description Language
UDDI	Universal Description, Discovery, and Integration

DBMS Database Management System

GUI Graphical User Interface

TEA TIBCO Enterprise Administration

Glossary

Message Simple data structure such as a string, a byte array, a record, or an object [3].

Service Business function, defined and implemented by an organization, that is exposed externally, or internally, so it can be reused [2].

SOA SOA is an architecture based on the notion that the assets of Information Systems in an organization are described and exposed as services. These services can be composed and orchestrated in business processes [1].

ESB The function of an Enterprise Service Bus is to route messages among services and provide functions such as data format transforming, reliable message routing, and service management [4].

Queue Queues, also known as channels, are logical pathways that connect the programs and convey messages. A channel behaves like a collection or array of messages, but one that is magically shared across multiple computers and can be used concurrently by multiple applications [3].

Producer-Consumer A producer, or sender, is a program that produces a message and sends it to a channel. A consumer, or receiver, is a program that receives a message by reading (and deleting) it from a channel [3].

Topic Communication channel utilized by the Publish-Subscribe paradigm [3].

Publish-Subscribe Communication paradigm that the producer of a message sends it to a topic and all the consumers of the topic receive the full message [3].

BP Business Processes are ordered set of business activities [5].

BPMN BPMN is a visual notation to create, share and improve knowledge on business processes [6].

XML XML is used to describe documents and data in a standardized, text-based format that can be easily transported via standard Internet protocols [8].

XPath xPath provide a standard syntax for accessing and manipulating the parsed nodes of an XML document [8].

SOAP SOAP enables transportation of Web Service calls and responses. SOAP is a messaging format that describes XML data according to W3C standards and represents call and response data in an envelop and message format [8].

UDDI UDDI is the directory standard for Web services, developed and maintained by the Universal Description, Discovery and Integration project, which is a part of the Organization for the Advancement of Structured Information Standards (OASIS) [8].

WSDL WSDL is based on the W3C XML standard, and describes what Web services are, what they do, and how they can be accessed to applications that want to access them via SOAP [8].

Web Services Web services can be defined as a method of integrating data and applications via XML standards across computing platforms and operating systems. Web-service enabled applications make calls and send responses to each other via SOAP. Web services are described to clients and other server applications by using WSDL which is associated with all standard Web services.Registration information and location of a Web service can be published to a UDDI directory, which is itself a Web service, and because it's a Web service, each UDDI registry server contains associated WSDL files and SOAP accessibility [8].

Framework Rogers [9] defined in 1997 that a framework is *a class library that captures patterns of interaction between objects. A framework consists of a suite of concrete and abstract classes, explicitly designed to be used together*. The definition that will be used in this dissertation will be the same but with a new set of properties (guidelines and best practices).

1

Introduction

Change is universal in life and in all organizations. The changes in organizations were accelerated by several factors like technological evolution, economy globalization, organizations productivity, increase on workers' knowledge, etc [1]. The most evident changes in the organizations are:

- **the interaction with the clients** have gone from face-to-face conversations to smartphones, e-mail, Internet, sms, etc.
- **business operations** were organized in departments (functional hierarchy) and now, to improve the intense and fast interaction between departments, they each have an information system.
- **business decentralization** made the organizations more and more distributed, e.g. distributed geographically. That implies that the information systems need also to be decentralized.
- **real-time management** is forced by the business agility, meaning that the management cannot be based on information updated quarterly or monthly.

In this time of rapid change, businesses needs to be **composable** so that capabilities can be projected and combined as they are needed for any internal or external system. A composable business drives more engaging applications and processes [2] because all the different components are closely working together, or better, all *integrated* [1].

Enterprise Integration is the creation of an IT architecture where all the processes, application, services and information systems collaborate to support the business goals. The most common integration architecture is the Service-Oriented Architecture, or SOA.

There is no formal definition for what is **SOA** but the following is among the most consensual: *"SOA is an architecture based on the notion that the assets of Information Systems in an organization are described and exposed as services. These services can be composed and orchestrated in business processes"* [1].

To fully understand the SOA concept, it is necessary to define the three fundamental aspects of SOA [2]. These are:

- **Service**: business function, defined and implemented by an organization, that is exposed externally, or internally, so it can be reused.
- **Service orientation**: A way of thinking about your business through linked services and the outcomes they provide.
- **Service-oriented architecture**: A business-centric architectural approach that is based on service-oriented principles.

The purpose of SOA is to **increase efficiency** for the integration of systems but, unfortunately, it is not easily implemented. By definition, enterprise integration has to deal with multiple applications running on multiple platforms in different locations, making the term ”simple integration” pretty much an oxymoron. The true challenges of integration span far across business and technical issues [3]:

- Enterprise integration requires a significant shift in corporate politics. Business applications generally focus on a specific functional area and not on the global business goal. Enterprise integration not only needs to establish communication between multiple computer systems but also between business units and IT departments.

- Because of their wide scope, integration efforts typically have far-reaching implications on the business. Once the processing of the most critical business functions are incorporated into an integration solution, the proper functioning of that solution becomes vital to the business.

- One important constraint of developing integration solutions is the limited amount of control the integration developers typically have over the participating applications.

- Despite the wide-spread need for integration solutions, only few standards have been established on this domain. The advent of XML, XSL and Web services certainly mark the most significant advance of standards-based features in an integration solution (XML Web Services standards address only a fraction of the integration challenges).

- The mix of technologies and the distributed nature of EAI(Enterprise Application Integration) solutions make deployment, monitoring, and trouble-shooting complex tasks that require a combination of skill sets. In many cases, these skill sets do not exist within the IT operations team or are spread across many different individuals.

There are no simple answers for enterprise integration. Usually, the issues that arise during development are solved by the most experienced developers, since they have solved a large number of them in the past. They understand the patterns of problems and associated solutions. They learned these patterns over time by trial-and-error or from other experienced integration architects [3].

The main goal of this work was to develop a **Service Integration Framework** using TIBCO ActiveMatrix® BusinessWorks™ 6 that implements a set of patterns, guidelines and best practices for the integration developers. We implemented a set of services that are always needed for any service, so that the developer doesn’t need to develop it every time, or a combination of naming with function structure to use in every situation.

The principal benefiters of this framework are the developers but by maintaining the consistent and cohesion in every projects all the organization benefits from the work developed.

In this document is presented some theoretical background, the design, use and evaluation of the integration framework, and the conclusions retrieved from the full experience.

2

Background

Contents

In this section it is presented some theoretical background about TIBCO, SOA and BPM (main technologies used in this document).

2.1 Company

This topic is presented in this section to give the reader a background on the company, how it was created and how achieve the position that it have today.

TIBCO Software Inc. is an American company, founded by **Vivek Ranadivé**, that develops software that enables the various applications, databases, and platforms used by companies to work together. TIBCO's products are known as integration software, the "middleware" that bridges the computer systems within a business and connects its software to outside suppliers, vendors, and customers [10].

Vivek Ranadivé left Bombay at age 17 with enough money to last two months. The year was 1958, and Ranadivé was headed to Cambridge, Massachusetts, where he had been accepted at the Massachusetts Institute of Technology (MIT). Ranadivé excelled in the academic world, earning an undergraduate degree in electrical engineering and computer science and a master's degree in engineering at MIT before crossing the Charles River to Allston, Massachusetts, where he earned a master's degree in business from Harvard Business School [10].

Ranadivé had a technological vision, but he was able to trim that vision to what was doable in the 1980s. Ranadivé's big idea, which he called "**The Information Bus**", was to have an interface for various types of software that would be as simple and modular as the interface for hardware. His ideas to provide fast access to information offered the biggest payoffs in the financial industry, and Ranadivé targeted it, developing a system that would give Wall Street traders integrated access to information from a wide variety of sources on a single workstation [11].

Ranadivé stayed on with the company, which was reincarnated as **TIBCO**, and continued to evangelize for his idea. The router manufacturer Cisco Systems was an investor and supporter of its products. Ranadivé became a Silicon Valley personality, in 1999 writing a book, "The Power of Now", enunciating his philosophy of the power of the real-time information and the information bus [11].

Figure 2.1: TIBCO Logo

2.2 SOA

This topic is presented in this section to give the reader a background on the term SOA and how it is used nowadays.

It is hard to find out who coined the term **SOA** but the first reports about it were published in 1996 by Gartner [13] analysts Roy W. Schulte and Yefim V.Natis [14].

Despite this, the term was created by Alexander Pasik, a former analyst at Gartner [13] for a class on **middleware** that he was teaching in 1994. Pasik was driven to create the term SOA because ”client/server” had lost its classical meaning. Many in the industry had begun to use ”client/server” to mean distributed computing involving communication between desktops. A desktop ”client” PC typically ran user-facing presentation logic, and most of the business logic. The back end ”server” computer ran the **DBMS**, and sometimes some business logic. To avoid confusion between the new and old meanings of ”client/server”, Pasik stressed ”server orientation” as he encouraged developers to design **SOA** business applications [14].

The real momentum for SOA was created by **Web Services**, which, initially driven by Microsoft, reached a broader public in 2000. Although Web Services do not necessarily translate to SOA, and not all SOA is based on Web Services, the relationship between the two technology directions is important and they are mutually influential: Web Services momentum bring SOA to mainstream users, and the best-practices architecture of SOA help make Web Services initiatives successful [14].

2.3 Web Services

XML 1.0 was standardized by the W3C in 1998, and by 2000, many IT vendors were publishing their version of Web Services. SOAP arose out of Microsoft’s COM technologies and was given to the W3C for standardization in May of 2000. WSDL came as a joint effort between IBM and Microsoft in March of 2001. After this, the parallel growth of both SOA and Web Services (both contributing to each other’s

popularity) took several years, and it has only been in the past few years that Web Services became a buzz-worthy topic [12].

These services have a set of previously defined desired properties [14] [17] that are displayed in the table 2.1.

Additionally to these desired properties, services need to be **loosely coupled**. Loosely coupled services are connected to other services and clients using standard, dependency-reducing, decoupled message-based methods [17]. When dependencies are minimized, modifications have minimized effects, and the system still runs when parts of it are broken or offline [14].

Property	Description
Self-Contained	All the definitions of SOA agree that it is a design goal that service be self-contained (independent, autonomous, autarkic) [14].
Coarse-Grained	Operations on services are frequently implemented to encompass more functionality and operate on larger data sets, compared with component-interface design [17].
Visible/Discoverable	To call a service, the client needs to know that the service exists. Often there is a public place where he can search for a service, and/or that describes all the details of a service [14].
Stateless	The ability of not maintaining business or technical state [14]. This means that the service needs to be viewed as a manager object that can create and manage instances of a type, or set of types [17].
Idempotent	The ability to redo an operation if its completion isn't completely certain [14].
Reusable	Ideally, in software development, each functionality should be implemented only once [14].
Composable	Services can use/call other services e.g. broader kinds of business functionality can be broken into smaller steps, which are themselves services. From a business point of view, the issue of composing services and decomposing business processes leads to business process modeling [14], which is presented in the next section.

Table 2.1: Desired Properties in Services

2.4 Microservices

Like SOA, Microservices are also considered a service-based architecture and, although they differ in architecture styles, they share many characteristics [15]. Microservices are an alternative approach to structuring applications by broken it into smaller and independent components allowing them to have greater scalability and availability [15] [16].

2.5 BPM

This topic is presented in this section because BPMN is the language used by TIBCO BusinessWorks™ to develop the functions.

Flexible and innovative business processes are one of the key elements that enable modern organizations to succeed. In order to define, share and improve knowledge on business processes, individuals need a standard way of describing them. The Business Processes Management Initiative (BPMI) put much effort to develop such a standard business process modeling notation - **BPMN** [6].

In 2001 BPMI began developing BPML (Business Process Modeling Language, an XML process execution language) and realized there was a need for a **graphical representation**. The individuals and vendors involved at the time decided a notation was required that was oriented toward the needs of a business user [7].

The Notation Working Group (who originally created BPMN within BPMI) was formed in August 2001. It was composed of 35 modeling companies, organizations and individuals who between them brought a wide range of perspectives [7].

2.6 Summary

To summarize, the important things to retain are, that TIBCO uses BPMN as main language to develop the company processes and can expose them as Web Services to other companies. The use of several Web Services combined can be classified as Service-Oriented Architecture.

3

Related Work

Contents

In this chapter it is presented relevant and related work. Initially it is presented the previous and current versions of TIBCO ActiveMatrix® BusinessWorks™, after that it is presented other integration solutions. Finally, it is presented the existing frameworks in the market (BusinessWorks 5 framework included).

3.1 Integration Solutions

3.1.1 TIBCO ActiveMatrix® Product Suite

TIBCO ActiveMatrix® product suite is a family of related products(Figure 3.1). This section provides an overview of these products [18] and a more detailed insight about BusinessWorks™.

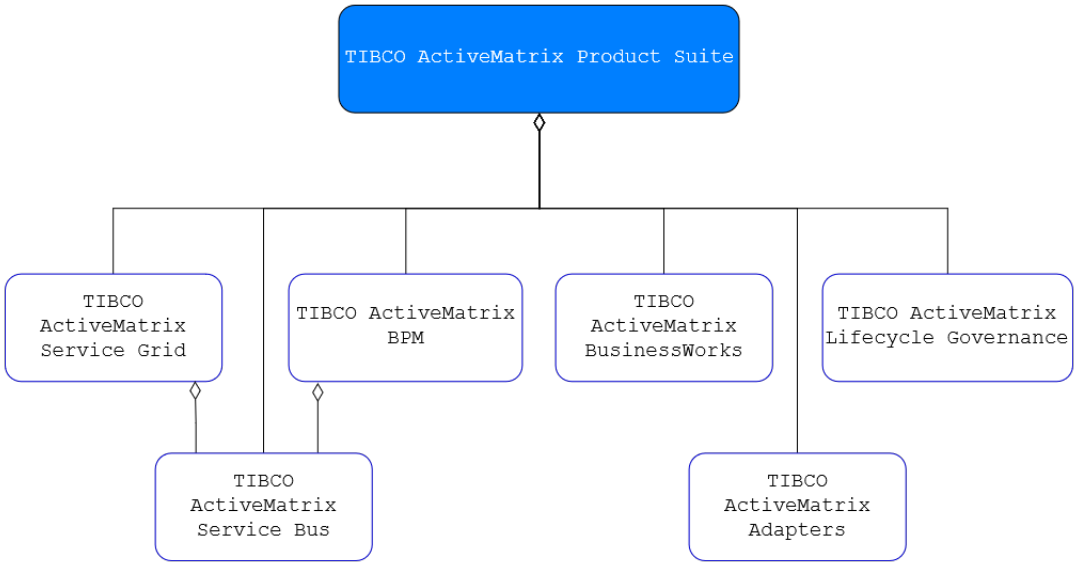


Figure 3.1: TIBCO ActiveMatrix Product Suite Architecture

The core of the suite is the **TIBCO ActiveMatrix® Service Bus**, which provides the foundational infrastructure for building solutions based on the service-component architecture (SCA) concept. **TIBCO ActiveMatrix® Service Grid** builds on this foundation, adding a number of implementation types that can be used for implementing components. **TIBCO ActiveMatrix® BPM** adds support for business process management. **TIBCO ActiveMatrix® Lifecycle Governance** framework provides lifecycle management, repository and registry capabilities to the enterprise. **TIBCO ActiveMatrix® Adapters** provide lightweight interfaces to many types of external systems [18].

The **TIBCO ActiveMatrix® BusinessWorks™** integration platform enables companies to rapidly

integrate systems and automate business processes. It supports all integration projects throughout the project lifecycle. It includes a common graphical user interface for adapter configuration, process design, and deployment. User administration, component management, and process monitoring are available via a web browser based GUI [20].

Next section presentes in more detail both versions of the product that currently are in use.

3.1.2 TIBCO ActiveMatrix® BusinessWorks™ 5

TIBCO ActiveMatrix® BusinessWorks™ is a scalable - it will remain effective when there is a significant increase in the number of resources and of users [19] - , extensible, and easy to use integration platform that allows the development of integration projects. It uses the TIBCO Designer GUI for defining Business processes and the TIBCO ActiveMatrix® BusinessWorks™ process engine executes the business process. The architecture is based on the following set of fundamentals [21]:

- Support for Standards
- Integrated Development Environment
- Extensible and Scalable

Support for Standards

A good integration platform must support standards for several reasons. A standards-based integration platform supports better as new applications are added to an enterprise or need to communicate with new business partners. Standards are essential while planning for the future of the project because standards facilitate updates. TIBCO ActiveMatrix® BusinessWorks™ 5 supports the most widely used standards for the different aspects of an integration project [21]:

- J2EE Compliant - JMS, EJB, JNDI
- Protocols - Web services (SOAP, WSDL), HTTP, HTTPS
- Messaging - JMS, TIBCO Rendezvous
- Data Description - Native support for DTD, XSD, and TIBCO AE Schema
- Data Representation and Expressions - Native support for XML, XPath
- Data Transformation - XSLT

TIBCO ActiveMatrix® BusinessWorks™ also supports a plug-in for B2B interactions.

Integrated Development Environment

With TIBCO ActiveMatrix® BusinessWorks™ all phases of the projects (design, deployment, and run-time environment) are tightly integrated, even though the run-time environment supports a distributed architecture [21].

1. At design time, you use TIBCO Designer to configure services, such as adapters, and design your business processes.
2. You can use the TIBCO Designer in test mode to debug the process definitions in your integration project.
3. You can use TIBCO Administrator to deploy processes to process engines and to deploy adapter services to adapters on the individual machines.
4. Optionally, you can configure manual activities using TIBCO Designer. When you deploy the project, that information is then used by InConcert. The users authorized to handle the manual activities can be specified in TIBCO Administrator, then exported to TIBCO InConcert.
5. You can start the processes using the TIBCO Administrator GUI. You can then monitor and manage all processes using TIBCO Administrator.

Extensible and Scalable

As a company grows, new applications are added and volume of data increases. TIBCO ActiveMatrix® BusinessWorks™ was designed to be extensible and scalable [21] in a way that it would keep up with the changes in any company.

Designer

The main feature that TIBCO ActiveMatrix® BusinessWorks™ contains is an easy-to-use drag and drop graphical development environment called TIBCO Designer. In image 3.2 is presented the view of the designer and its panels.

The project panel shows a project example that consists of resources that contain the functionality needed for the integration system. This includes services and any business logic that may be applied to that information. A resource corresponds to an object in a TIBCO application, such as an FTP activity, a process definition, or a specific adapter instance. Context-sensitive palettes organize resources into related groups. Which palette is displayed depends on the currently opened resource and on the developer preferences. The developer can drag and drop resources from the palette into the design panel to

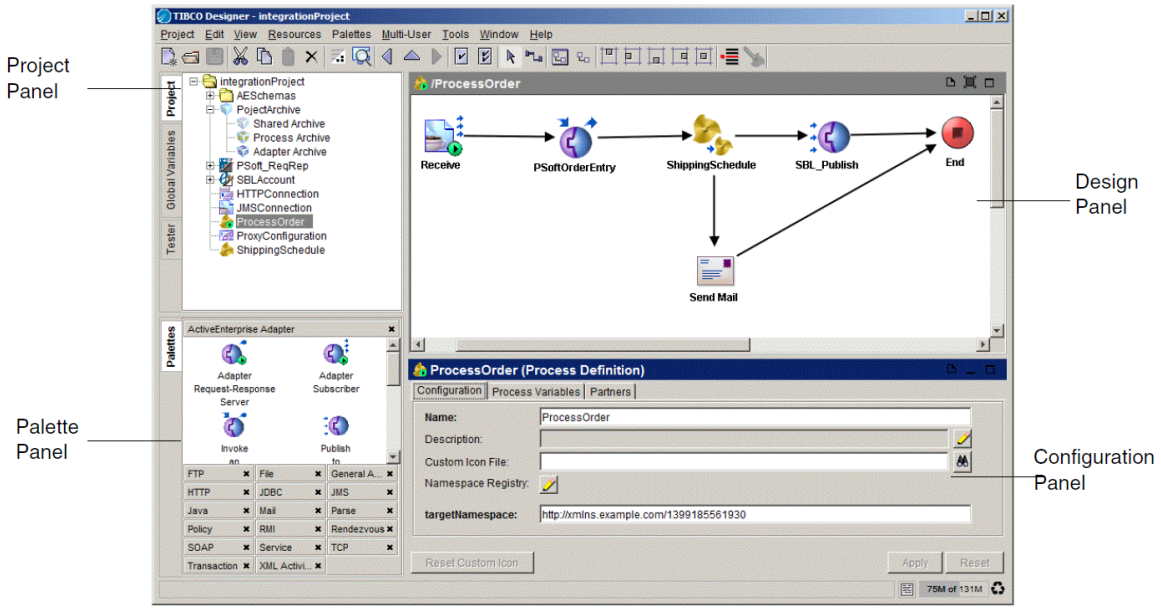


Figure 3.2: TIBCO Designer

add them to their project [21].

The Enterprise Archive resource allows the creation of an Enterprise Archive file (EAR file) that can be used to deploy the project. The EAR file contains shared archives and process archives that were specified by the developer. These archives contain the adapter configurations and process definitions wished to deploy. TIBCO Administrator can use the EAR file to create a deployment configuration [21].

Messaging

TIBCO ActiveMatrix® BusinessWorks™ also have a messaging system that can reliably handle the volume of messages that will be sent and received [21]. The system have the following characteristics:

- **Guaranteed delivery and fault tolerance** - If a message cannot be delivered because the recipient was unavailable, the messaging system must queue that message and continue to operate. The queued message must then be redelivered to the recipient when it become available.
- **Distributed architecture** - A distributed, loosely coupled system is much more likely to support the fault-tolerance.
- **High throughput** - The company business cannot afford performance degradation at the time when the work load increases.

- **Scalability** - The message system must support the scalability of business growth, change and integration with other departments.

TIBCO ActiveMatrix® BusinessWorks™ is based on messaging standards with proven track records. Supported protocols include TIBCO Rendezvous, JMS, and HTTP [21].

Adapters

Adapters translate information into the appropriate format. They receive information from a source application and publish it to the business process in a shared format [21].

Business Process Modelling

The business processes describe the actual flow of data inside the enterprise. In Designer is possible to design and test all the implemented processes. Image 3.3 shows a simple process that is part of the example used in this section.

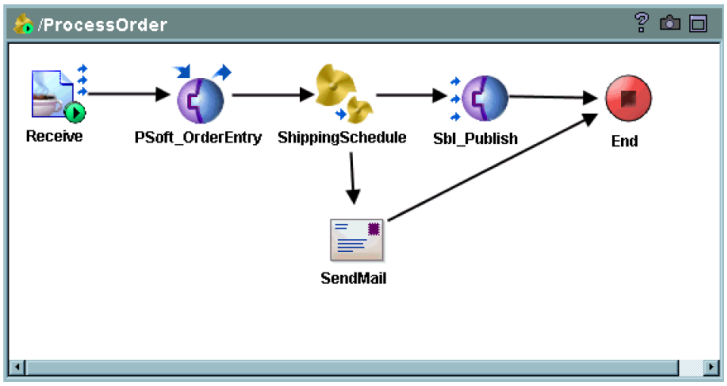


Figure 3.3: TIBCO Process

Deployment and Administration

The administration of the applications is done through a tool called Administrator. With this tool is possible to configure all variables, running services, etc.

This tool can be leveraged by the naming rules, the definition of shared and non shared variables, etc.

The deploy is done by creating an EAR, or Enterprise Application Resource, uploading it to the Administrator and creating an instance to run the developed code.

Project Life Cycle

TIBCO BusinessWorks™ has pre-defined development phases that are presented in the Fig. 3.4.

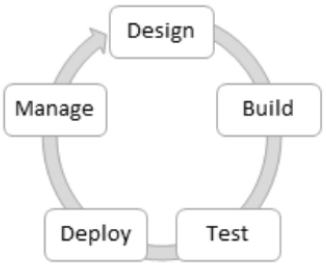


Figure 3.4: Development Phases

These phases are sufficient to a project lifecycle, but most projects go through more stages than the ones presented.

Initially, a project is born when a business case emerges in a organization. After that it needs to be analyzed and specified. The next stage is when it is developed. When it is complete, it needs to be tested. The test stage is divided in several levels since there is different test levels (unitary tests, integrated tests). After that, it needs to be certified by a third party organization. The last stage is testing with the final clients and when it is over it enters in production.

3.1.3 TIBCO ActiveMatrix® BusinessWorks™ 6

This section will describe in more detail the product TIBCO ActiveMatrix® BusinessWorks™ 6, why it is used, what it allows the developer to do and what benefits provides.

The product was developed by TIBCO and allows the developer to easily connect applications, whether on-premises or in the cloud, providing major reliability and scalability. This SOA and integration platform is one of the most efficient on the market. It has an easy-to-use drag-and-drop graphical development environment and allows the deployment of applications in its own runtime environment [22].

Studio

TIBCO Business Studio (Fig.3.5) provides a workbench that is used to create, manage, and navigate resources in Eclipse workspace. A workspace is the location, on the developer machine, where the artifacts related to BusinessWorks™ projects are stored [23].

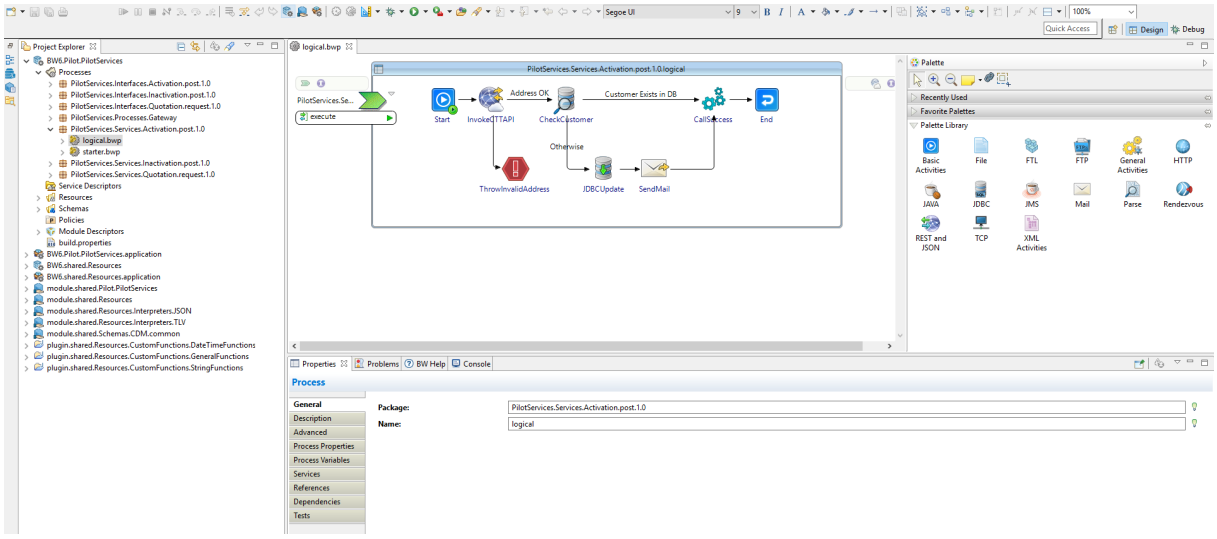


Figure 3.5: TIBCO Studio

The user interface of the TIBCO Business Studio is very similar to the TIBCO Designer interface. This aspect was maintained to facilitate the migration of TIBCO ActiveMatrix® BusinessWorks™ users from version 5 to 6.

Applications

An application is a collection of one or more modules and can range from simple to very complex. An application contains one application module (defined in the next subsection) and zero or more shared modules. A process that is responsible for initiating the business logic at runtime is used to implement a component in an application module [25]. Fig.3.6 presents all the application elements.

Once an application is implemented, the developer can either run or debug directly in the studio, or generate a deployable artifact that can be deployed later in the runtime environment. [25]

Modules

A module is an Eclipse project that is configured for BusinessWorks™. There are two types of modules:

- **Application Module** - normally contains several resources like processes, service descriptors or schemas [25].
- **Shared Module** - application used by multiple projects. There is also an extension that is called a Binary Shared module. This module is a normal shared module but with the particularity that the implementation details are hidden from the developer [25].

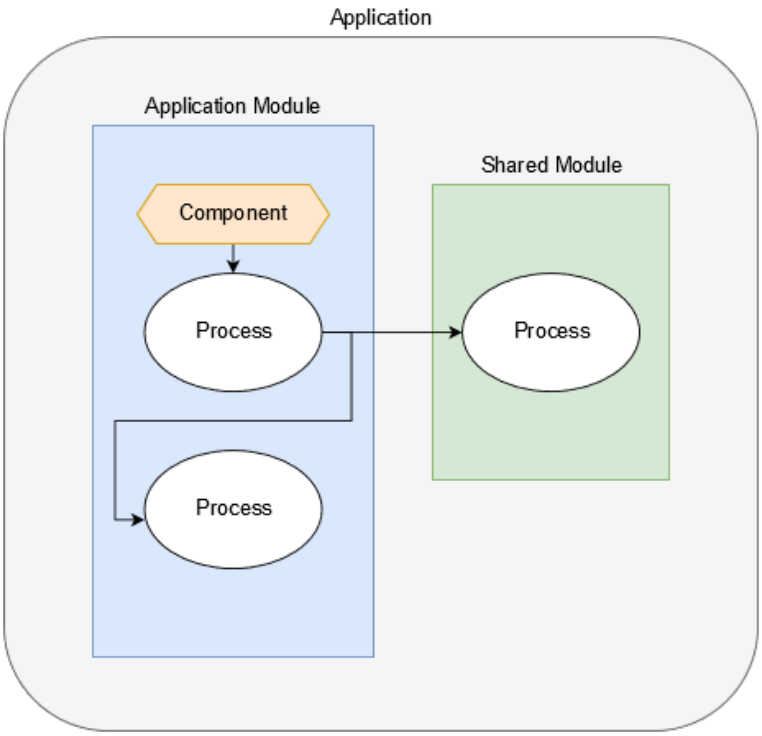


Figure 3.6: Application Elements

Processes

Processes capture and describe the flow of information in an enterprise between different data sources and destinations. Processes are comprised of activities that accomplish tasks. The flow of data between activities in a process is represented using transitions, conditions and mappings [25].

Activities

Activities are the individual units of work in a process. They generally interact with an external system and perform a task. Activities that perform similar tasks are grouped in a pallet [25].

Activities can be classified in three types:

- Regular Activities: perform a specific task, can have input and output and can also state the faults they can throw at runtime.
- Process Starter Activities: configured to react to events. They trigger the execution of a process when the event occurs. For example, the HTTP Receiver activity starts a process when a HTTP request is received.
- Signal-in Activities; special activities that proceed with executing the process instance when an appropriate asynchronous event is received.

Transitions

Transitions represent the flow of execution from one activity, or group, to another. They are represented as arrows and are unidirectional [25].

Transitions fall into one of the following categories:

- Transitions without Conditions: control automatically flows from one activity, or group, to the next without any conditions.
- Transitions with Conditions: the transition is only taken if the conditions previously defined are met.
- Error Transitions: special transitions that specify the activities or groups to execute in case of an error.

Shared Resources

Shared Resources are resources that contain common configuration data that can be referenced from multiple places. The developer can define a shared resource and then reference it from multiple activities in the same or different process. For example, a JDBC Connection resource can be defined and used in any of the JDBC activities [25].

There is also the possibility to share variables. They are used to define data for modules and jobs. There are two types of shared variables:

- Job Shared Variables: used to share data within a job. At runtime, the engine allocates a new variable for each job and the value of that variable is not visible outside the job.
- Module Shared Variables: used to share data across all processes in a module. The module shared variable is visible to all process instances within the same module.

The difference is that when a job expands across module boundaries, a job shared variable is visible outside the module it was set in, while the module shared variable is visible only inside the module in which it was set [25].

Architecture

The architecture of TIBCO ActiveMatrix® BusinessWorks™ 6 is presented in image 3.7.

- **AppSpace** - Logical group of one or more AppNodes (group of one or more applications). When deploying an application to an AppSpace it will be deployed to all AppNodes which are part of that

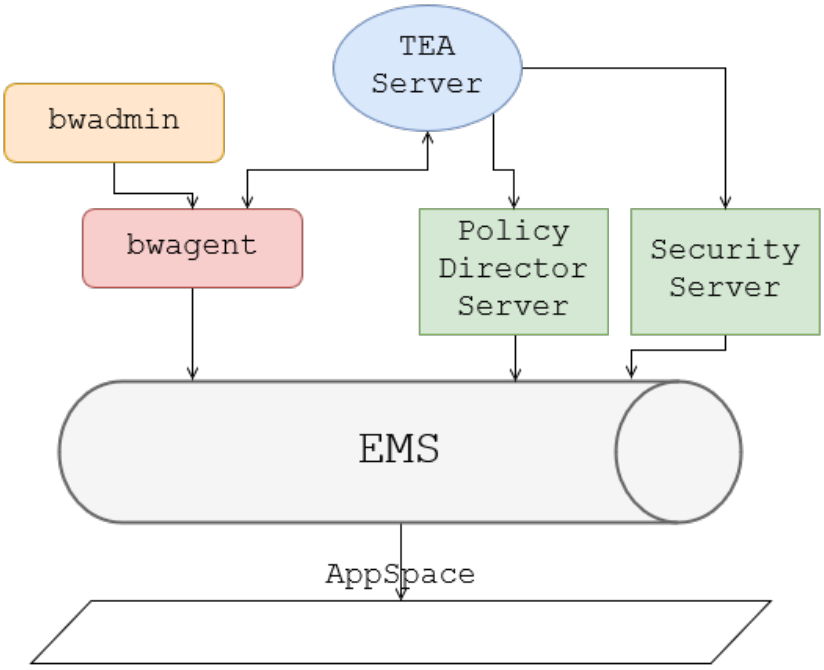


Figure 3.7: BusinessWorks™ 6 Architecture

AppSpace. In previous versions of BusinessWorks™ this concept did not exist, the only logical grouping of applications was the domain [25].

- **Administration** - Applications are deployed into runtime environments and managed using a centralized administrative interface called TIBCO Enterprise Administrator, or TEA. With this tool is possible to perform common administrative tasks such as authenticating and configuring runtime artifacts [25]. Fig.3.8 displays an overview of the administration architecture.

  To deploy an application is necessary to wrap it in an application archive, or EAR. The application archive is the deployment unit.

  When the applications are deployed is required to define in which domain, AppSpace and AppNode it will be stored. A domain is a logical group that provides an isolated environment for applications and their resources to reside.

  Inside the domains can exist multiple AppSpaces that are a group of one or more AppNodes. Fig 3.8 presents this architecture [25].

- **bwagent** - Process that exists in each installation. This process is responsible for provisioning the AppNodes and enabling the communication agents that are located in different machines [25].

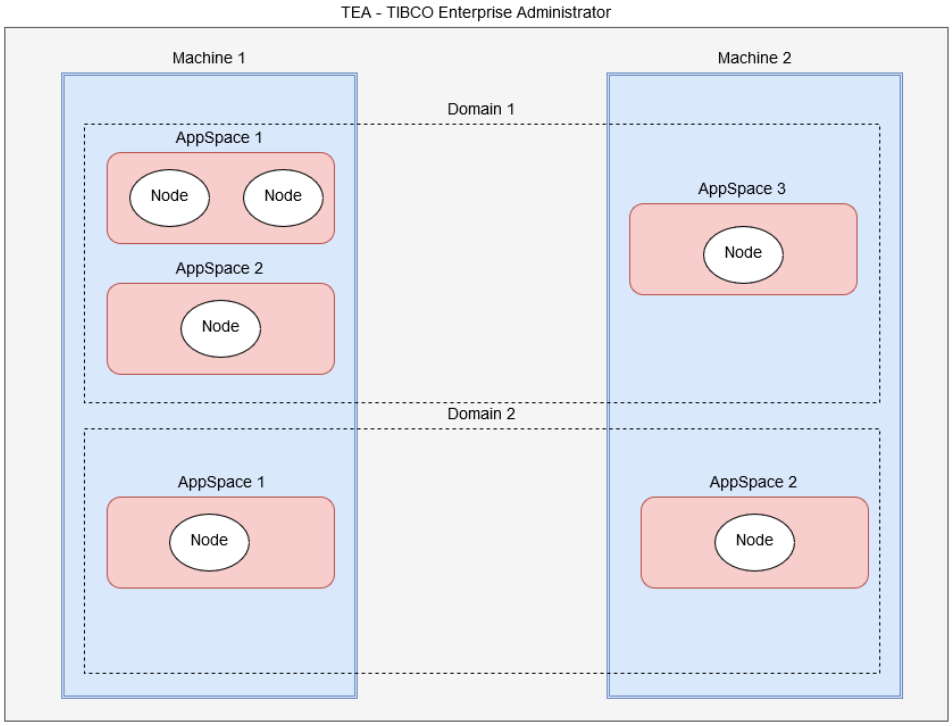


Figure 3.8: Administrative Architecture

- **bwadmin** - Tool that provides a command line console that can be used in local mode or enterprise mode to create and manage domains, AppSpaces, AppNodes, archives, and applications [25].

- **EMS** - TIBCO Enterprise Message Service™ software allows the developer to send messages from their applications in a format that conforms to the Java Message Service (JMS) specification. The software also extends the JMS specification with a reliable delivery mode and a no-acknowledge acknowledgement mode [24].

Runtime Concepts

The major components of the runtime environment are:

- AppNode: is an operating system process that hosts and executes one or more Applications. Is the actual JVM which is equivalent to what was called bwengine in previous BusinessWorks™ versions [25].

- Process Instance: the execution of any process creates an execution scope for the activities that are a part of the process [25].

- Job: execution of a component. Each job has a unique id referred to as JobId. [25]

All the applications developed in this TIBCO product are Java applications. When they are deployed a Java Virtual Machine, or JVM, runs with the application on it.

TIBCO ActiveMatrix® BusinessWorks™ 6 is established in four major pillars: open architecture, flexibility, modularity and support for standards [25]. These pillars are also the most important benefits that the product can provide:

- Flexibility: the product is designed to make adding, upgrading and swapping of business components easy. This is demonstrated by the zero coding model that allows to select and drop activities and configure them, by the ability to build tightly or loosely coupled services or by the encapsulation of configuration data (minimal configuration properties exposed by the application) [25]

- Openness and Extensibility: public APIs that allow the development of custom activities and XPath functions, extensible OSGi, or Open Service Gateway Initiative, based runtime or extensible administration framework based on TIBCO Enterprise Administrator [25]

- Modularity: large teams and distributed development through modular constructs or reusability with a consistency model across different technologies (Processes, java classes, XSDs, etc) [25]

- Standards-based Data: representation and transformation (XML, JSON, XPath, XSD, XSLT), protocols and APIs (SOAP, JSON, WSDL, HTTPS and JMS) and FTP, JNDI and TCP [25]

BusinessWorks™ Upgrade

TIBCO released this new version of BusinessWorks™ in 2014, to go along with the new era of API and Mobile predominance. Because the end goal is different, the product itself also is.

Organizations are no longer just integrating internal ERP, SaaS, custom or legacy applications; today they're exposing the data that fuels their mobile applications, web channels and open APIs. This data allows them to engage in real-time discussions with customers [26].

3.1.4 Other Integration Solutions

Besides TIBCO, exists other platforms in the market for the integration development. Despite the fact that **TIBCO was already chosen**, it is an interesting to research these solutions to give a different perspective of the market and widen the knowledge base. This section presents some of these solutions and their comparison with TIBCO.

Mulesoft Anypoint Platform

This open-source integration solution is developed by Mulesoft [28] and allows designing, building and managing the projects lifecycle. It is divided in 6 layers presented in Fig.3.9 and has a visual studio that is an Eclipse-based integration development environment [27]. With that in mind the developers can use the studio to connect all applications and data, design and publish APIs using RAML [29], deploy their projects, with Mule runtime, and much more [27] [28].

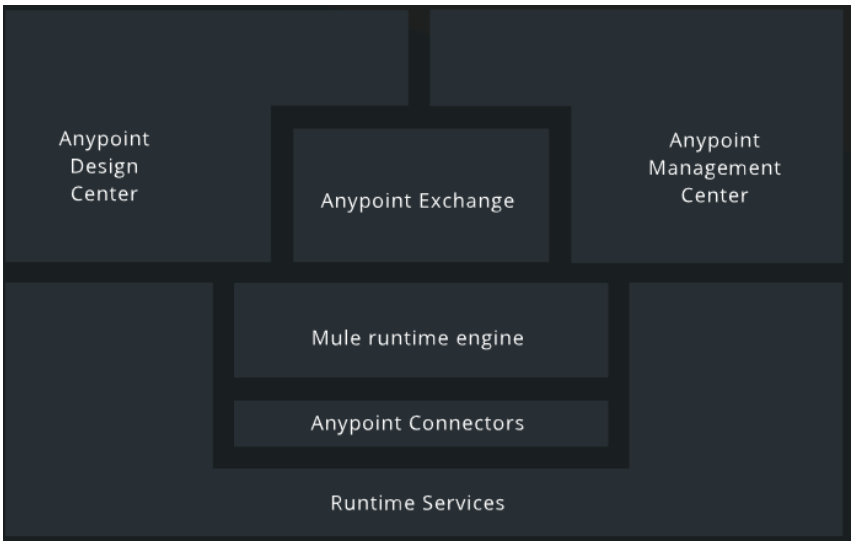


Figure 3.9: Mulesoft Anypoint Layers

Microsoft BizTalk

Microsoft BizTalk Server is an Inter-Organizational Middleware System (IOMS) that enables companies to automate business processes, through the use of adapters which are tailored to communicate with different software systems used in an enterprise. Created by Microsoft, it provides enterprise application integration, business process automation, business-to-business communication, message broker and business activity monitoring [30].

Oracle Suite

Oracle also provides an integration solution that is divided in SOA suite and BPM Suite.

Oracle SOA Suite (Fig.**??**) provides a complete set of service infrastructure components for designing, deploying, and managing composite applications. Oracle SOA Suite enables services to be created, managed, and orchestrated into composite applications and business processes. Composites enable

you to easily assemble multiple technology components into one SOA composite application [31].

The Oracle BPM Suite provides an integrated environment for developing, administering, and using business applications centered around business processes [32].

IBM Integration Bus

IBM Integration Bus provides flexible integration services that offer universal connectivity between any two or more end points. It also provides a comprehensive run time for interpreting, transforming, and routing various message formats. The use of a common toolset, powerful transformation techniques, and adapter technologies provides a flexible integration engine that can be deployed anywhere with integration services that are developed by using a common skill set [33].

Some of the main integration scenarios that are enabled by IBM Integration Bus are shown in Figure 3.10.

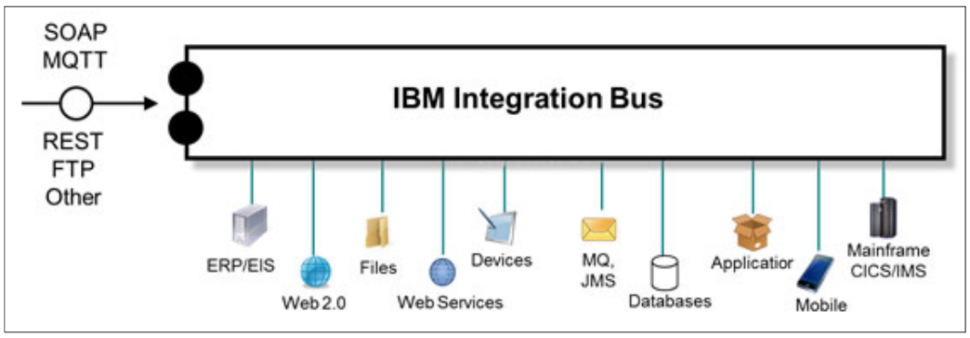


Figure 3.10: IBM Integration Bus

3.2 Integration Frameworks

All frameworks must have a set of basic characteristics that make them valuable and usable. The first one is that it should be **extensible**, that means, have the possibility to grow. After that, it should be **reusable** - well documented and easy to use. It must also be **safe**, so that the developer cannot destroy it. Finally, it must be **complete** and **efficient**.

An integration framework can be used to integrate different systems, technologies, applications and products without the need of repeat code because patterns are already defined. The main goal is for the

developer to focus only on the functional requirements of the project.

The framework brings a lot of advantages like great flexibility, the fact that can be attached as simple library into a project, it's open source and reduce the development complexity [34].

3.2.1 CODIT Framework

The first integration framework is from a company called Codit and is composed by a wide set of components. It aids to resolve common integration problems by applying different patterns. When these components are used they allow the developer to implement less code which, in turn, reduce drastically the number of tests [35].

The framework provides an wide set of reusable components to BizTalk Server, WCF, WF, Azure BizTalk Services and Integration Cloud and it process over ten million messages every day in different clients of several industries [35].

3.2.2 Oracle WebLogic Framework

The integration framework provided by WebLogic Integration offers a standards-based architecture for hosting application views: business-oriented interfaces to enterprise applications [36].

Instead of accessing an enterprise information system by directly invoking it, the developer can simply edit the application views, create new application views, or delete obsolete ones. This layer of abstraction, formed by application views, makes it easy for non-programmers to maintain the services and events exposed [36].

If the developer defines a "Customer Management" application view for a CRM (Customer Relationship Management) system, then it is likely to add only services and events that are related to customer management. Application views can be customized for a specific business purpose, they work much better than the "one size fits all" approach used by many other EAI systems [36].

3.2.3 TIBCO ActiveMatrix® BusinessWorks™ 5 Framework

A framework is already defined for the previous version of TIBCO ActiveMatrix® BusinessWorks™. This section will present it in more detail.

This framework can be divided in two separate approaches: the **objective** part, where a set of processes, sub-processes, schemas, resources and configurations are defined; and the **subjective** part where the good practices and guidelines used to implement the projects in a consistent manner are passed to the developers. To better understand the framework concepts it will be presented a set of running examples.

The framework is well set in five major pillars: Transport, Functions, Logging, Naming and Catalog.

Transport

The message transport task in the existing framework is handled by a simpler adaptation of an ESB. It implements a communication system between mutually interacting software applications in a Service-Oriented Architecture.

The used adaptation is named **Enterprise Message Service**, or EMS. It is the TIBCO implementation of Java Message Service which provides queue and topic APIs to send messages [24]. In Fig. 3.11 a conceptual diagram to define the EMS architecture is presented.

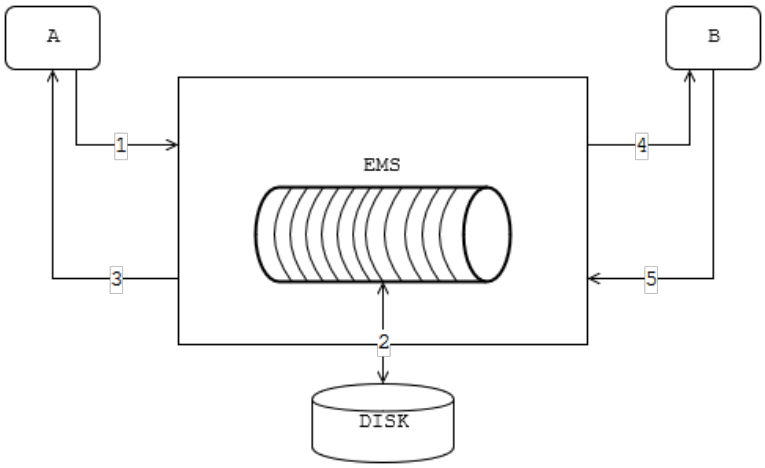


Figure 3.11: EMS Architecture

The messages are sent to a queue, or publish to a topic, and the listener is ready to consume them when they arrive to the queue or topic. The system has a persistence mechanism that saves in disk the unconsumed messaged.

All the messages are standard XML because it has a set of features that helps the developer:

- Simplicity;

- Easy to read (for humans and for machines);
- No limits to the creation of tags;
- Focus on the information structure and not on its appearance.

All the functions defined using the framework use the EMS to communicate internally. In the Fig. **??** is presented an example of a Process Starter.

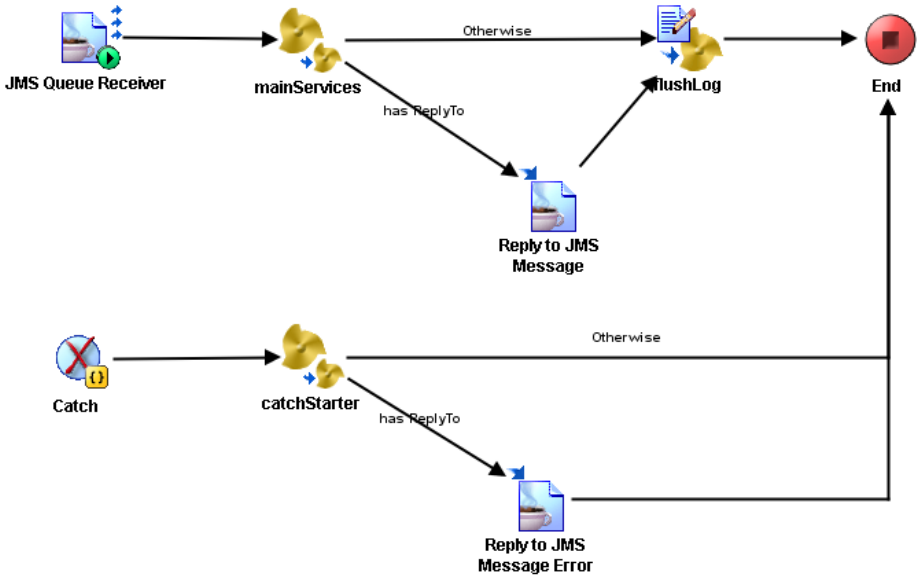


Figure 3.12: Process Starter

Functions

There are three type of functions in the existing framework: Services, Processes and Interfaces.

Services are business functions, defined and implemented by an organization, that are exposed externally, or internally, to the company [2]. They use the standard protocol and message format in the company middleware.

Processes are a set of activities that interact to achieve a result [5]. They do not need to follow a protocol or a message format, but they must be easily integrated with services.

Interfaces are abstractions for the services and processes implementations. Normally, they are used when client applications can not use the standard protocol or message format.

Logging

In here, as in all software applications, there must be a log to keep track of all that is done in the project.

The Central Log is essential a set of three tables (Header, Keys and Data) that have all the necessary information about the data, business keys and process/services/interfaces features.

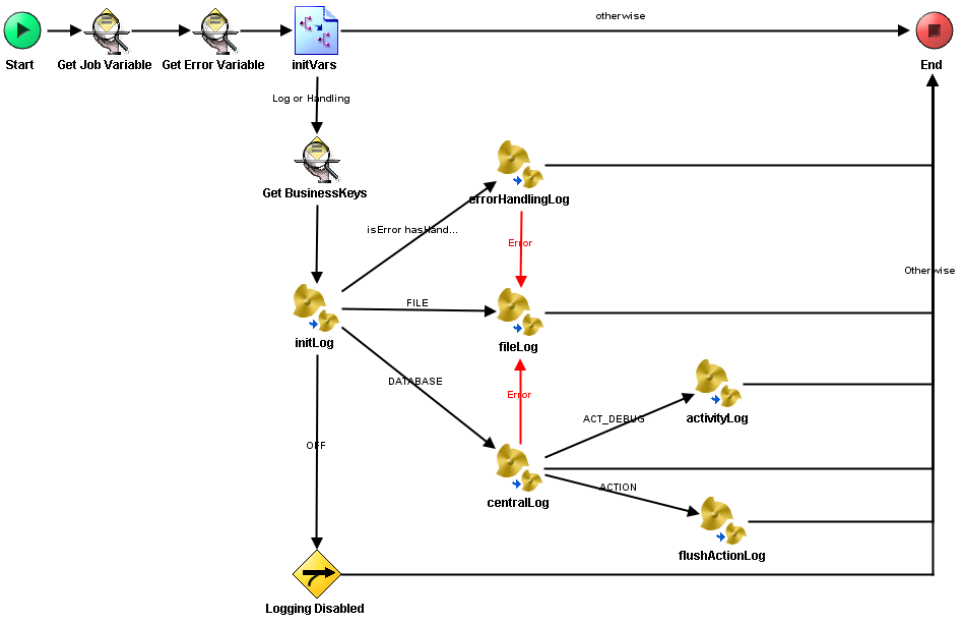


Figure 3.13: flushLog process

The services, processes and interfaces write in the Central Log by calling a process called FlushLog(Fig.3.13). This process sends the messages, using the centralLog subprocess, through a set of Database Adapters, or ADBs, to the databases. These adapters work like the interfaces in TIBCO, they simply transform the data, that TIBCO processes sends, to a format that the database can understand.

Naming

The naming of the resources is very important to maintain consistency and coherence throughout all the projects.

All the packages maintain the same logic of naming: Domain.Service.Name.Operation. No punctuation is allowed, apart from period. Underscores and dashes must be avoided, the name must be in upper camel case and the operation in lower camel case. Is possible to have a subprocess folder inside

the package described above.

Also, all the global variables packages must follow the same logic of naming. The only difference is that the actual name of the variable must be in upper camel case.

The main benefit that this rules brings to the developers is the easy reading of the project organization and its documents.

Catalog

The Catalog is a set of tables designed to facilitate the developer work and to maintain every aspect of the project organized. It is divided in three types:

- Service Catalog: a single table where all the services are registered. It has a lot of columns with information about each service, for example, if it is active, if has error handler or what is its profile.
- Configurations Catalog: a set of tables that describe all configurations. Some examples are Error Handler, Properties or Routing.
- Error Categorization: is a single table that translate every native code to the correspondent canonical code.

4

Solution Architecture

Contents

In this section will be presented the solution developed along the course of this work. The solution is vaguely based on the BusinessWorks™ 5 framework to facilitate its use among the developers that already known the previous version. For a better understanding of its concepts it will be presented the ”Catch” process as a running example along the several elements of the framework.

4.1 Architecture

The BusinessWorks™ 5 framework is designed to have a shared module that have all the logic of the framework “attached” to each developed application. On the other hand, the architecture defined for the new framework was a bit different.

The framework developed, as shown in Fig.4.1, is divided in two parts: the **APPSF** - Application Service Framework - that have all the logic of the framework processes and several **Shared Modules** (one for each application in that running in that AppSpace). This shared module is simply a set of interfaces that connect by an endpoint reference to the application framework to hide the implementation from the clients.

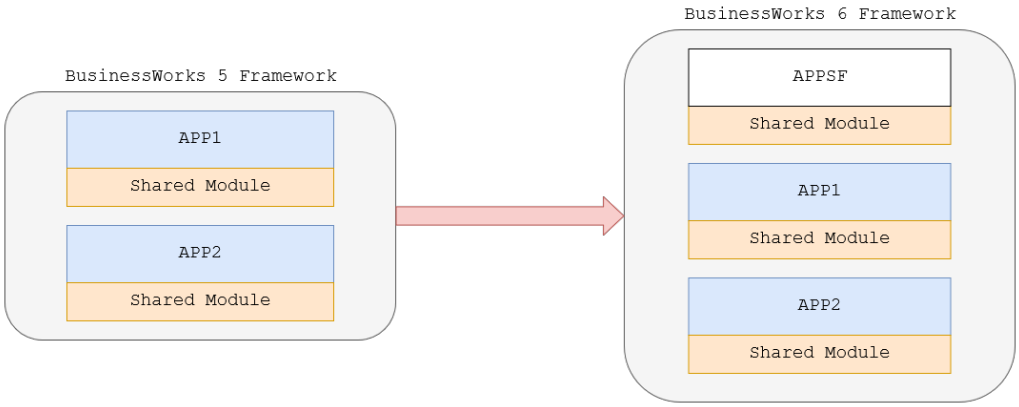


Figure 4.1: Architecture Framework Update

This change was decided in order to maintain the **framework logic transparent** to the developers and to **improve the performance** of the running applications. The performance was improved because the necessity to run the framework processes in each application ended. With this change the processes provided by the framework were used and treated like a web service of another application.

In Fig. 4.2 it is possible to see the APPSF, that was defined with the name ”BW6.shared.Resources”, and the Shared Module. In the application it is possible to see the developed services, like the flushLog or the Catch, and in the shared module the processes used to communicate between every application

and the framework application.

4.2 Functions

This aspect didn't suffer any significant change. There still exist three types of functions: Processes, Interfaces and Services and they are used in the same manner as in the BusinessWorks™ 5 framework.

All Interfaces and Services are divided in Logical and Starter. The Starter (Fig. 4.3) just receive the request by the EMS, call the Logical, answer to the request and register the transaction in the log. The Logical (Fig. 4.4) have the actual logic of the function (because the example used in the images is an interface it has very simple logical process).

Whenever any of this functions returns an error it will be caught like an exception in Java. In Fig. 4.3 is possible to see the area where the exception will be treated. In there all functions should call the "Catch" process. With the definition of this process by the framework the developer can almost ignore the exception treatment (except if a special treatment should be implemented).

4.3 Transport

The transport and interchange of messages will maintain the use of **TIBCO Enterprise Message Service** infrastructure because it is a well known, robust and reliable product. Since the new version was released to accompany the API and Mobile era, the framework was developed to allow the exchange of messages using **JSON or XML** format.

Another difference that is visible is the way that the shared modules communicate with the application framework. It was used an activity that create an endpoint reference (SetEPR) [37] in all the processes of the shared module (Fig.4.5). This reference connect to the services defined in the application framework through a WSDL, like a normal Web Service. These services have the same characteristics of the services defined in the previous section so they also are separated between logical (Fig.4.6) and starter.

4.4 Logging

The logging pillar had a major change because all the architecture behind the Databases has changed.

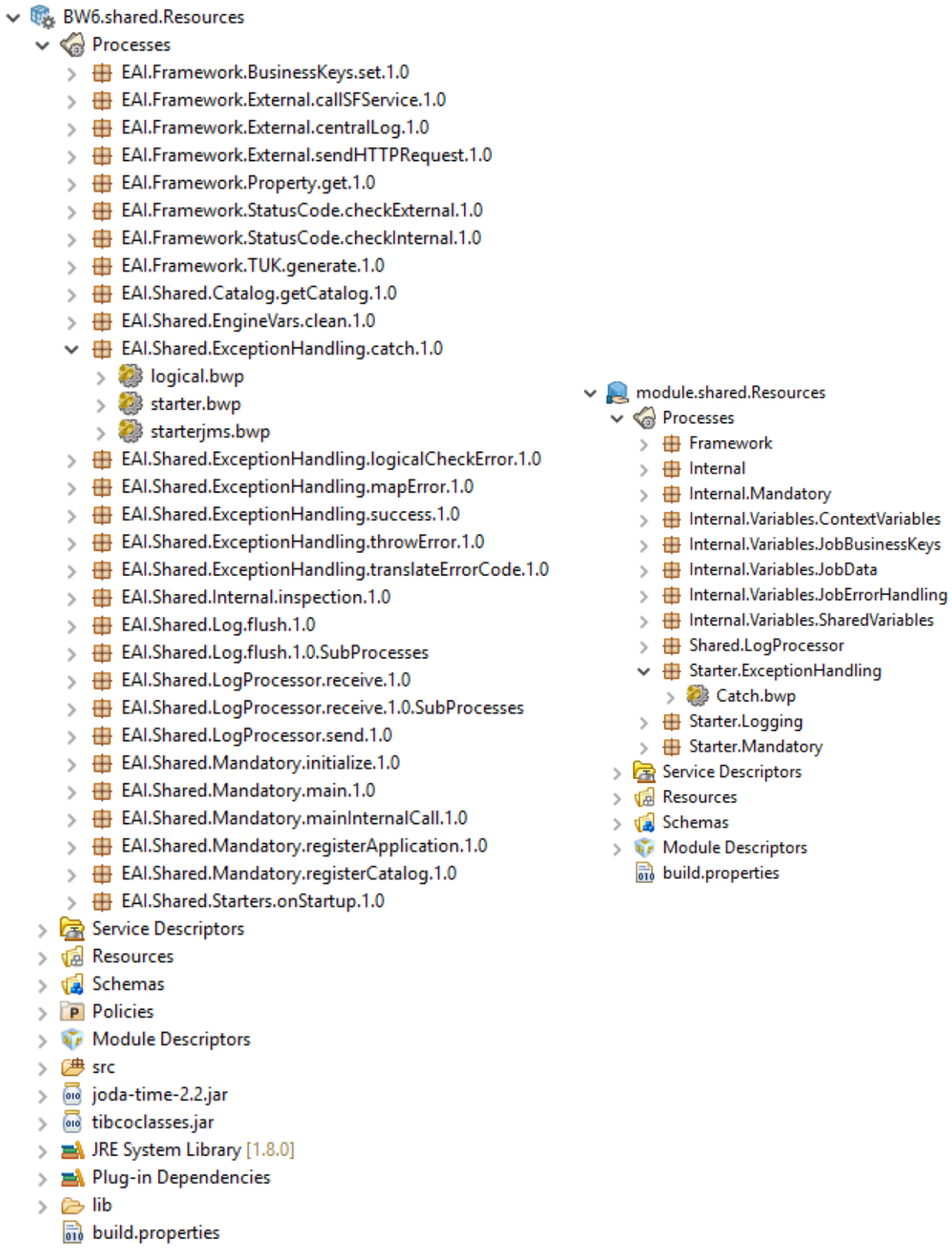


Figure 4.2: Framework Implementation

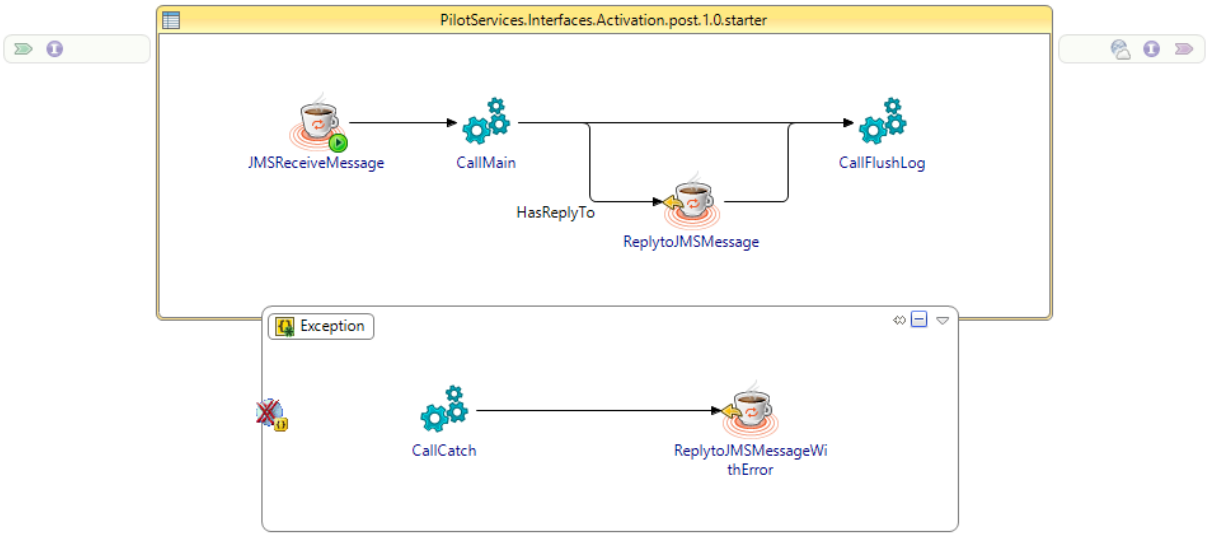


Figure 4.3: Interface/Service Starter

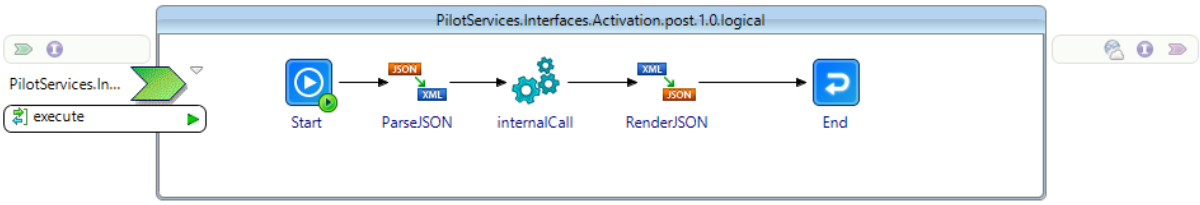


Figure 4.4: Interface/Service Logical

The initial framework has a relational model (Fig. 4.7) between the three databases (Header, Keys and Data). This aspect changed to an Analytical Model, or a star schema (Fig. 4.8). A star schema is a database organizational schema that has a table in the center called Fact table that has all the other tables connected to it. The fact table has as attributes the ID as Foreign Key of all the other tables.

This change was made in order to, once more, improve the performance of the system.

In the document initially presented it was referenced that the update of the ADBs to the version 7 was being considered. This change was not implemented because it didn't offer the necessary performance to accommodate the amount of transactions that the company does daily. Therefore it was decided to implement the same analytical model in Oracle.

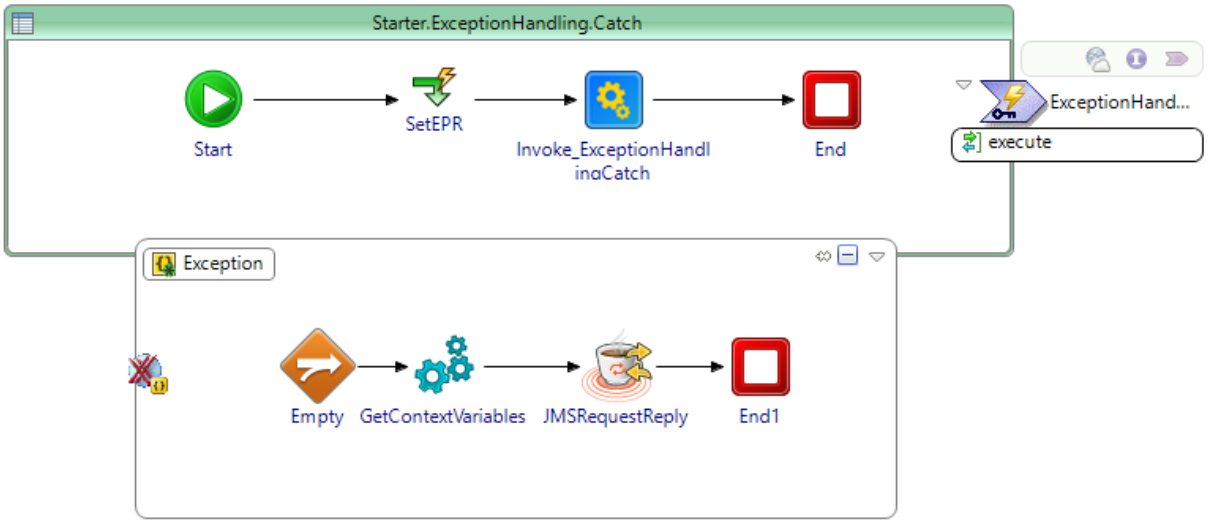


Figure 4.5: Catch Process

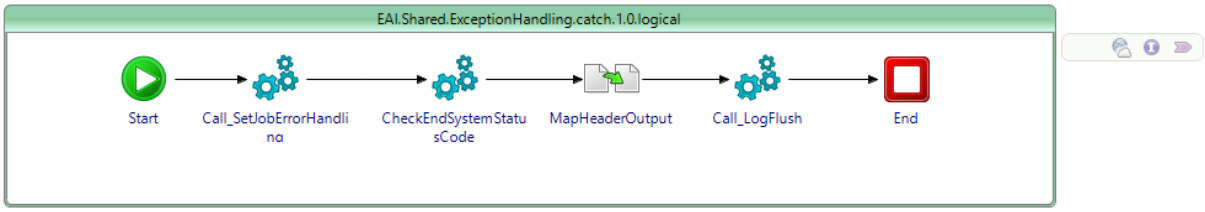


Figure 4.6: Catch Logical

4.5 Naming

This is one of the subjective aspects of the framework, so it is a good practice that the developers should follow.

The naming conventions had some changes. The functions names were improved, as well as their namespaces. This was necessary because the use of an endpoint reference needed to have very well define namespaces and names.

4.6 Catalog/Project Lifecycle

Even though the organizational schema of the databases will change, the tables will remain the same, therefore the Catalog pillar will remain the same as in the previous version.

Like the Catalog pillar, also the Project Lifecycle will remain the same as in the previous version.

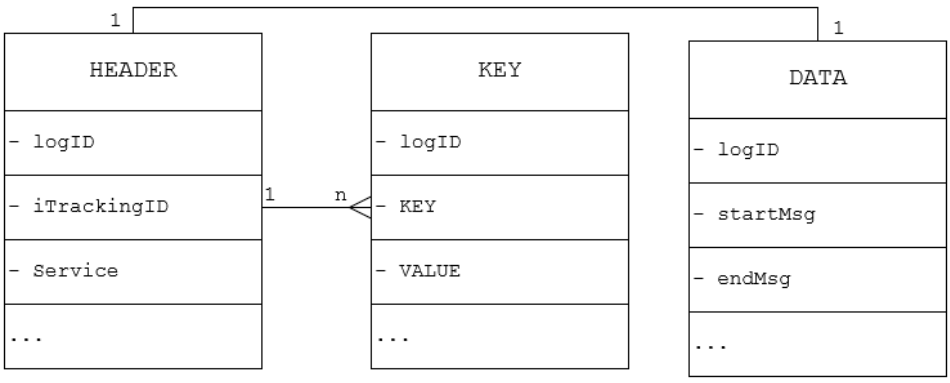


Figure 4.7: Logging Relational Model

4.7 Deployment and Administration

The administration now will be done in TIBCO Enterprise Administrator. With the evolution of BusinessWorks™, TIBCO developed a new tool to facilitate administration of the projects.

TEA, as was previously defined, provides a centralized administrative interface to manage and monitor multiple TIBCO products deployed in an enterprise.

It was implemented a deployment protocol called Blue&Green that defined that there are two different AppSpaces, one for deploy and another for runtime (Fig.4.9 (a)). When the developer deploys an application it always go to the deploy AppSpace, while the previously deployed application is kept running in the runtime AppSpace (Fig.4.9 (b)). This allows the organization to provide high availability to the client and chose, along with the client, when is the best time to upgrade to the newer version of the code.

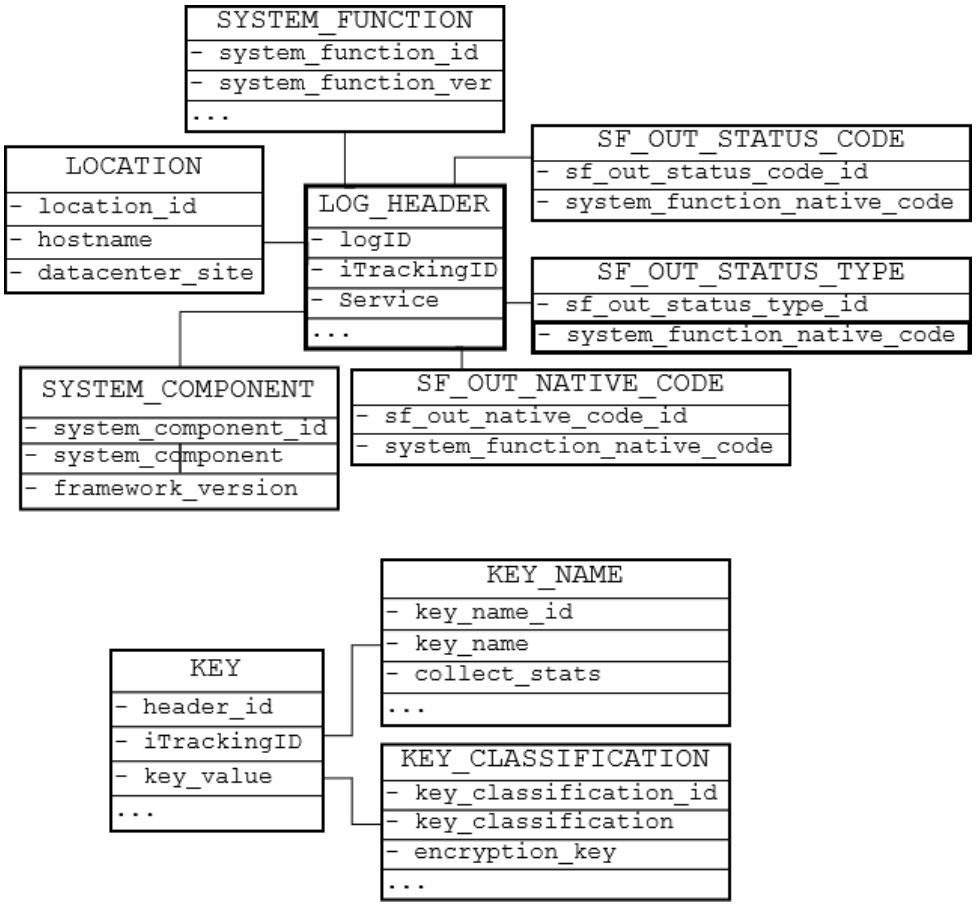
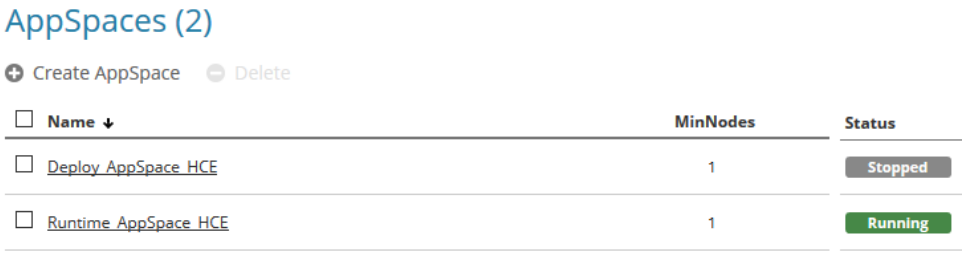


Figure 4.8: Logging Analytical Model



(a) Deploy and Runtime AppSpaces

(b) Deployed Applications

Figure 4.9: Framework Deployment Protocol

5

Work Evaluation Methodology

Contents

This chapter presents an assessment methodology designed for SOA-based systems. Not a lot of research has been done about the evaluation of SOA systems so it was necessary to find some way to do it.

A system, to achieve the status of high quality, needs to be evaluated and improved. Our solution takes advantage of a series of metrics that can be directly retrieved by the system. The use of metrics allows us to quantify the system in mathematical terms and identify where it can be improved. We separated the assessment process in three phases [38]: Metrics Definition, Metrics Application and Result Analysis.

5.1 Metrics Definition

Since these systems are inserted in the scope of software engineering the way that we start to define the metrics was by using **ISO 9126**. This is a standard that can be used to describe the quality of software systems [39].

We started by considering the following six attributes ”functionality”, ”reliability”, ”usability”, ”efficiency”, ”maintainability” and ”portability” [38] but, as it will be explained, some of them weren’t suitable for the required assessment.

5.1.1 Identifying Design Quality Attributes

”Portability” and ”reliability” were eliminated because they would vary depending on the SOA solution instead of the design. We also removed ”usability” because the designs of SOA systems are not directly related with the interaction with a user. Finally, the attribute ”functionality” was also rejected because there weren’t pre-defined system functional requirements for this work.

Since the assessment was more directed to the design of the solution the attributes ”efficiency” and ”maintainability” were respectively replaced with ”effectiveness” and ”understandability” [38]. Efficiency and maintainability can be related to the manner that a software application respond and because we want to evaluate more than just that this change seemed suitable.

Two additional quality attributes - **”reusability”** and **”flexibility”** - were identified and added. These qualities were added to satisfy an important feature on the SOA system, the ability to provide loosely coupled services [38].

The resulting set of quality attributes are described in Table 5.1.

Quality Attribute	Description
Effectiveness	The degree of business requirements reflected in the design.
Understandability	A measure of the effort necessary to learn or comprehend the design.
Flexibility	The ease of changing the previous design to accommodate new functionalities.
Reusability	Measures how much of the design allows reapplication to other solutions.

Table 5.1: Target Quality Attributes

5.1.2 Linking Design Components to Quality Attributes

The SOA design properties are derived from the components of the SOA-based system like services, operations or messages. A system that is considered well designed exhibit a set of features represented by the design properties identified in Table 5.2

Design Property	Description
Coupling	The strength of dependency between services in system
Cohesion	The strength of relationship between operations in a service
Complexity	Measures the difficulty of understanding relationship between services
Design Size	The size of the system design
Service Granularity	The appropriateness of size of services
Parameter Granularity	The appropriateness of size of parameters
Consumability	The likelihood of other services to discover the given service

Table 5.2: SOA Design Properties

The identified characteristics are measured using four group of metrics that are defined according to their role and scope [38]. These groups are the service internal metrics, service external metrics, system metrics and a group of derived metrics from the previous groups.

- Service internal metrics, represented in Table 5.3, use service internal elements like activities, operations, message features and service name.

Metric	Description
SIM_NO	Number of Operations
SIM_NFPO	Number of Fine-Grained Parameter Operations
SIM_NMU	Number of Message Used
SIM_NAO	Number of Asynchronous Operations
SIM_NSO	Number of Synchronous Operations
SIM_NINO	Number of Inadequately Named Operations

Table 5.3: Service Internal Metrics

- The second group of metrics in Table 5.4 are service external metrics and use information from the connected services. It allows to measure the features of consumer and produces services.

Metric	Description
SEM_NCSL	Number of Consumers in Same Level
SEM_NDPS	Number of Directly Connected Producer Serivces
SEM_NDCS	Number of Directly Connected Consumer Services
SEM_NTPS	Total Number of Producer Services
SEM_NTCS	Total Number of Consumer Services

Table 5.4: Service External Metrics

- The third group, in Table 5.5, measure the characteristics of the system in general.

Metric	Description
SM_SSNS	System Size in Number of Services
SM_NINS	Number of Inadequately Named Services
SM_NINO	Number of Inadequately Named Operations
SM_TMU	Total Number of Message Used
SM_NAO	Number of Asynchronous Operations
SM_NSO	Number of Synchronous Operations
SM_NFPO	Number of Fine-Grained Parameter Operations
SM_NPS	Number of Process Services
SM_NIS	Number of Intermediary Services
SM_NBS	Number of Basic Services

Table 5.5: System Metrics

- The final group makes use of the three previously defined metrics groups. Metrics in this group measure design properties. Table 5.6 describes the derived metrics group and their relationship with the design properties defined in the previous section.

After the calculation of the design properties it is possible to calculate the quality attributes using the formulas defined in table 5.7 [38].

5.2 Metrics Application

5.2.1 POC Comparison

In this chapter is presented all the metrics defined in the previous chapter for two identical **POC - Proof of Concept** - projects. One using the integration framework develop in this work - **SFW (Service Framework)** - and other without any framework - **NSFW (Not Service Framework)**. After that, it is

Metric	Formula	Design Property
Average Number of Directly Connected Services (DM_ADCS)	$\frac{SEM\_NDPS + SEM\_NDCS}{SM\_SSNS}$	Coupling
Inverse of Average Number of Used Message (DM_IAUM)	$\frac{SM\_SSNS}{SM\_TMU}$	Cohesion
Number of Operations (DM_NO)	$SM\_NSO + SM\_NAO * 1.5$	Complexity
Number of Services (DM_NS)	SM_SSNS	Design Size
Squared Avg.Number of Operations to Squared Avg. Number of Messages (DM_AOMR)	$\frac{(\frac{SM\_NAO+SM\_NSO}{SM\_SSNS})^2}{(\frac{SM\_TMU}{SM\_SSNS})^2}$	Service Granularity
Coarse-Grained Parameter Ratio (DM_CPR)	$\frac{SM\_NSO + SM\_NAO - SM\_NFPO}{SM\_NSO + SM\_NAO}$	Parameter Granularity
Adequately Named Service and Operation Ratio (DM_ANSOR)	$\frac{SM\_SSNS - SM\_NINS}{SM\_SSNS * 2} + \frac{SM\_NSO + SM\_NAO - SM\_NINO}{(SM\_NSO + SM\_NAO) * 2}$	Consumability

Table 5.6: Derived Metrics

Quality Attribute	Formula
Effectiveness	$0.33 * Cohesion + 0.33 * ServiceGranularity + 0.33 * ParameterGranularity$
Understandability	$-0.66 * Coupling + 0.25 * Cohesion - 0.66 * Complexity - 0.66 * DesignSize + 0.25 * ServiceGranularity + 0.25 * ParameterGranularity + 0.25 * Consumability$
Flexibility	$-0.22 * Coupling + 0.61 * ServiceGranularity + 0.61 * ParameterGranularity$
Reusability	$-0.5 * Coupling + 0.5 * Cohesion + 0.5 * ServiceGranularity + 0.5 * Consumability$

Table 5.7: Quality Attributes Formulas

possible to analyze if the work done was beneficial for a TIBCO developer.

Because the framework is a significantly sized and complex application, the POC developed was relatively simple, as it is possible to see in the image 5.1, to facilitate the assessment of the solution. It has three interfaces, three services and a gateway process. The logic is that a request arrive at the gateway process and it is send, via EMS, to the interface that is specified in the URI. The interface transforms the

JSON input in a XML and calls the respective service to do the specified logic.

For the **application with the framework**, the developer only has to implement the interfaces, the services and the internal calls (processes located in module.shared.Pilot.PilotServices used to invoke the service logic by the interface) using the shared module of the framework to perform the operations outside the business logic - an example is presented in Annex A. Also, he only has to implement the resources required for the business logic and not for the logic of the message interchange or the database connection.

We can also see in Figure 5.1 that the **application that didn't use the service framework** is a much more complex one - an example is presented in Annex B. Because it didn't use the framework the developer had to implement all the logic and all the resources that the project required, like the exception handling or the logging writing.

Because of the complex difference between these two applications the time required for the development of each one are very different. The time required to develop the application with the framework is much lower than the time needed to develop a full project without framework.

On top of that, the know-how that the developer needs to have to implement a project without the framework is significantly higher than the one required to implement a project with the framework.

Finally, the guidelines and good practices implemented side by side with the framework development also aids the developer to maintain coherence and consistency along the course of the project. After the project is finished another developer or manager can analyze the work done and can easily navigate through the application.

5.2.2 Metrics Results

In this section we present the metrics calculated using the method defined in the previous chapter.

We start by analyzing the developed applications and try to retrieve the internal, external and system metrics. After that, it is possible to calculate the derived metrics by using each respective formula. Each derived metric can be directly mapped to a design property. It is possible to also calculate the quality attributes previously defined.

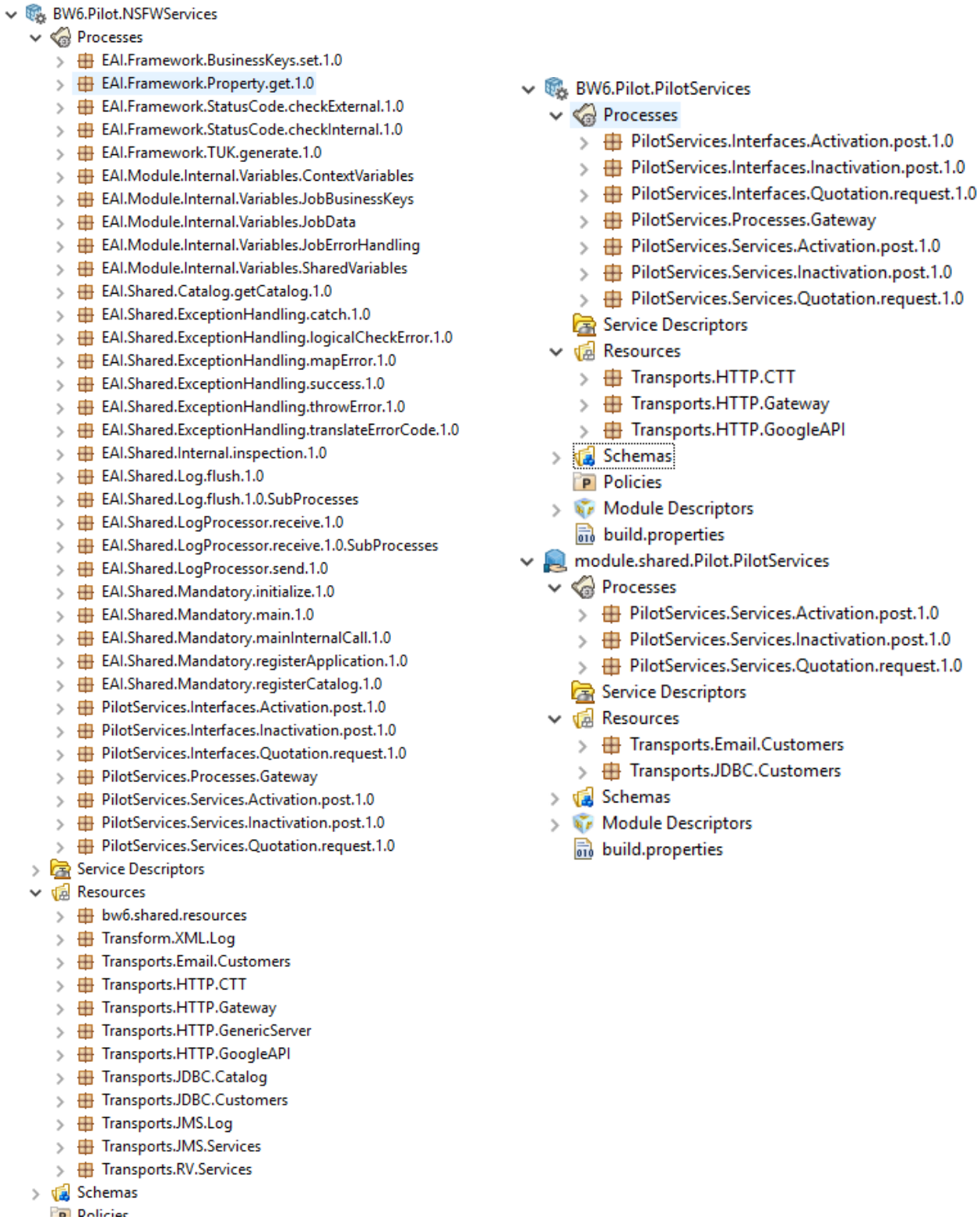


Figure 5.1: NSFW POC - SFW POC

Internal Metrics

Table 5.8 presents the internal metric results obtained in each POC application.

Metric	Description	NSFW	SFW
SIM_NO	Number of Operations	27	6
SIM_NFPO	Number of Fine-Grained Parameter Operations	14	0
SIM_NMU	Number of Message Used	24	12
SIM_NAO	Number of Asynchronous Operations	0	0
SIM_NSO	Number of Synchronous Operations	27	6
SIM_NINO	Number of Inadequately Named Operations	2	2

Table 5.8: Service Internal Metrics Values

External Metrics

Table 5.9 presents the external metric results obtained in each POC application.

Metric	Description	NSFW	SFW
SEM_NCSL	Number of Consumers in Same Level	7	7
SEM_NDPS	Number of Directly Connected Producer Services	0	0
SEM_NDCS	Number of Directly Connected Consumer Services	3	3
SEM_NTPS	Total Number of Producer Services	1	1
SEM_NTCS	Total Number of Consumer Services	6	6

Table 5.9: Service External Metrics Values

System Metrics

Table 5.10 presents the system metric results obtained in each POC application.

Metric	Description	NSFW	SFW
SM_SSNS	System Size in Number of Services	3	3
SM_NINS	Number of Inadequately Named Services	0	0
SM_NINO	Number of Inadequately Named Operations	1	1
SM_TMU	Total Number of Message Used	24	12
SM_NAO	Number of Asynchronous Operations	0	0
SM_NSO	Number of Synchronous Operations	27	6
SM_NFPO	Number of Fine-Grained Parameter Operations	14	0
SM_NPS	Number of Process Services	21	1
SM_NIS	Number of Intermediary Services	0	0
SM_NBS	Number of Basic Services	0	0

Table 5.10: System Metrics Values

Derived Metrics

Table 5.11 presents the derived metric results obtained in each POC application.

Metric	Formula	Design Property	NSFW	SFW
Average Number of Directly Connected Services (DM_ADCS)	$\frac{SEM\_NDPS + SEM\_NDCS}{SM\_SSNS}$	Coupling	1	1
Inverse of Average Number of Used Message (DM_IAUM)	$\frac{SM\_SSNS}{SM\_TMU}$	Cohesion	0.125	0.25
Number of Operations (DM_NO)	$SM\_NSO + SM\_NAO * 1.5$	Complexity	27	6
Number of Services (DM_NS)	SM_SSNS	Design Size	3	3
Squared Avg.Number of Operations to Squared Avg. Number of Messages (DM_AOMR)	$\frac{(\frac{SM\_NAO+SM\_NSO}{SM\_SSNS})^2}{(\frac{SM\_TMU}{SM\_SSNS})^2}$	Service Granularity	1.266	0.25
Coarse-Grained Parameter Ratio (DM_CPR)	$\frac{SM\_NSO + SM\_NAO - SM\_NFPO}{SM\_NSO + SM\_NAO}$	Parameter Granularity	0.481	1
Adequately Named Service and Operation Ratio (DM_ANSOR)	$\frac{SM\_SSNS - SM\_NINS}{SM\_SSNS * 2} + \frac{SM\_NSO + SM\_NAO - SM\_NINO}{(SM\_NSO + SM\_NAO) * 2}$	Consumability	0.536	0.639

Table 5.11: Derived Metrics Values

5.2.3 Quality Attributes

Table 5.12 presents the quality attributes results obtained in each POC application.

5.3 Result Analysis

This section presents a brief analysis of the results obtained with the assessment of the developed solution. The results are presented using a graphical representation of the data to facilitate the analysis and its perception.

Quality Attribute	Formula	NSFW	SFW
Effectiveness	$0.33 * Cohesion + 0.33 * ServiceGranularity + 0.33 * ParameterGranularity$	0.907	0.743
Understandability	$-0.66 * Coupling + 0.25 * Cohesion - 0.66 * Complexity - 0.66 * DesignSize + 0.25 * ServiceGranularity + 0.25 * ParameterGranularity + 0.25 * Consumability$	-19.858	-6.065
Flexibility	$-0.22 * Coupling + 0.61 * ServiceGranularity + 0.61 * ParameterGranularity$	0.846	0.543
Reusability	$-0.5 * Coupling + 0.5 * Cohesion + 0.5 * ServiceGranularity + 0.5 * Consumability$	0.463	0.069

Table 5.12: Quality Attributes Values

To present palpable results that demonstrate the advantages provided by the framework utilization, it was necessary to develop an application that did used the solution (SFW) and another one that didn't (NSFW). With both applications we were able to calculate the values for the design properties and quality attributes defined in chapter 5.

These values show us that the usage of the framework bring a lot of advantages to the developer that is implementing the application but can also cause some troubles.

A big advantage of using the framework is the huge complexity reduction. As it is possible to see in Fig. 5.2, the complexity value was reduced from 27 to 6, that means a reduction of around 78%. Other advantages brought by the framework are the increase of cohesion and consumability, and also the reduce of service granularity. The reduction on the service granularity implies that each service developed using the framework has less connections and requests to other services and can be easily decomposed.

A small disadvantage that can be retrieved by the presented results is the increase on the parameter granularity.

It is also observable in Fig. 5.2 that the parameters Coupling and Design Size do not change, but this is a direct consequence of the size of the application developed.

According to the Fig. 5.3, that presents the values of the quality attributes, the main advantaged retrieved from the framework utilization is the Understandability of the application. This was expectable to happen because of the decrease on the Complexity value obtained in the Fig. 5.2.

The remaining quality attributes presents a better view on the application that did not use the frame-

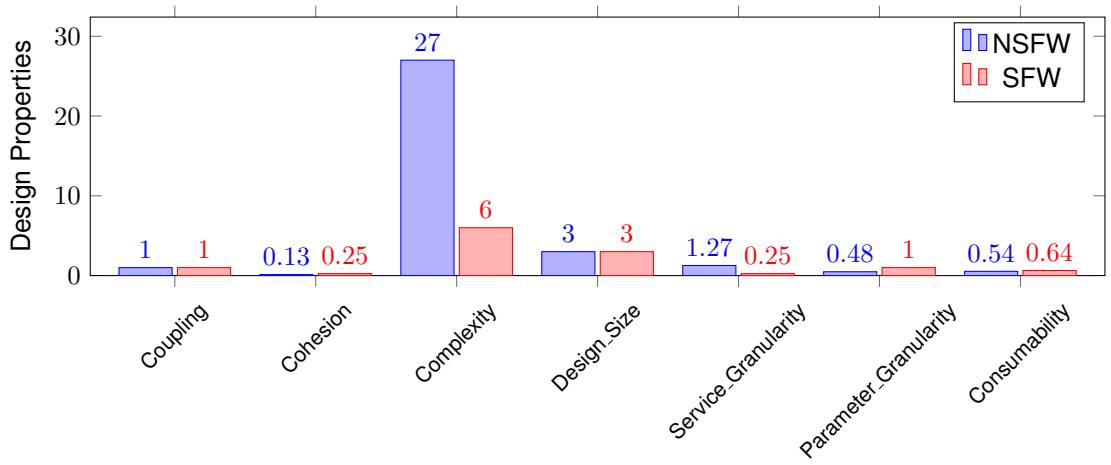


Figure 5.2: Design Properties Values Comparison

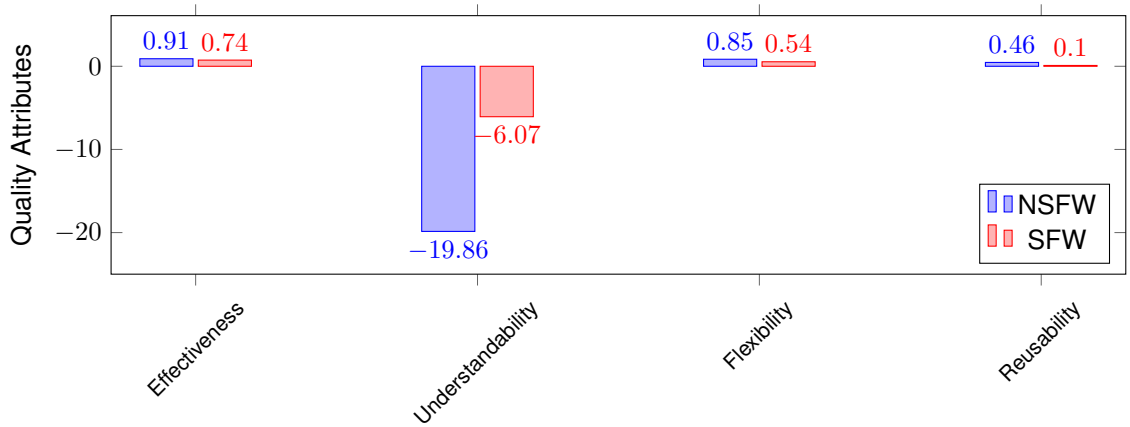


Figure 5.3: Quality Attributes Values Comparison

work. The Effectiveness, Flexibilty and Reusability on the application without the framework are higher that on the application with the framework.

These values were expected, since all the logic of the framework is based on the application without the framework, therefore, all the small services and subprocesses, that needed to be developed, enter these statistics.

In conclusion, it is possible to understand that the usage of the framework provides more advantages than disadvantages and that it contributes to facilitate the life of the developer, since it will take less time and effort to develop the projects.

6

Conclusion

Contents

Change is universal in life and in all organizations. SOA enables us to handle rapid business changes with more flexibility and feasibility. SOA is a share-as-much-as-possible architecture pattern that places heavy emphasis on abstraction and business functionality reuse.

With the creation of the latest version of BusinessWorks™, it was necessary to develop a service framework that could be reused as much as possible by the developers to maintain consistency and cohesion through all the projects.

With that in mind, it was presented in this thesis, a solution to accommodate these requirements. The solution was developed in 8 stages: Architecture, Functions, Transport, Logging, Naming, Catalog, Project Lifecycle and, Deployment and Administration.

The evaluation method was used to assess two applications, one using the framework (SFW) and another not using it (NSFW). With the results obtained by this assessment we could conclude that the use of the solution developed was beneficial in a series of ways.

6.1 Future Work

The following tasks are proposed for future work:

- Evaluate the framework using the expert feedback from developers. This will include providing the framework to a set of developers, let them use it for a pre-determined amount of time and use questionnaires and interviews to receive their feedback.
- Expand the framework. Increase the amount of features that the framework can provide with the help of the evaluation made to the developers in the previous point.
- Present the developed solution to TIBCO Software Inc.so they can analyze it, improve it and distribute it to their clients.

Bibliography

[1] Beth Gold-Bernstein, William Ruh 2004 Addison-Wesley: Enterprise Integration: The Essential Guide to Integration Solutions - ISBN 978-0321223906

[2] IBM Enterprise: Integration Throughout and Beyond the Enterprise 2014 https://www.redbooks.ibm.com/redbooks/pdfs/sg248188.pdf

[3] Gregor Hohpe, Bobby Woolf 2003 Addison-Wesley: Enterprise Integration Patterns: Designing, Building, and Deploying Messaging Solutions - ISBN 978-0321200686

[4] J. Yin and Z. Wu and H. Chen and C. Pu and S. Deng 2009: A Dependable ESB Framework for Service Integration doi.ieeecomputersociety.org/10.1109/MIC.2009.26

[5] Desel, Jorg and Pernici, Barbara and Weske, Mathias: Business Process Management: Second International Conference, BPM 2004, Potsdam, Germany, June 17-18, 2004, Proceedings Springer Science & Business Media

[6] Mariusz Momotko, Bartosz Nowicki 2003: Visualisation of (Distributed) Process Execution based on Extended BPMN http://ieeexplore.ieee.org/document/1232036/

[7] White, S.A. and Miers, D. 2008 : BPMN Modeling and Reference Guide: Understanding and Using BPMN - ISBN 978-0977752720

[8] Brian Benz, John R. Durant John Wiley & Sons, May 7, 2004: XML Programming Bible

[9] Sherif M. Yacoub, Hany Hussein Ammar : Pattern-oriented Analysis and Design: Composing Patterns to Design Software Systems (2003)

[10] Company Profile, Information, Business Description, History, Background Information on TIBCO Software Inc. http://www.referenceforbusiness.com/history2/96/TIBCO-Software-Inc.html

[11] Bassett, R. 2009 Harvard University Press: The Technological Indian - ISBN 978-0674504714

[12] Footen, J. and Faust, J. 2008 Elsevier/Focal Press: The Service-oriented Media Enterprise: SOA, BPM, and Web Services in Professional Media Systems - ISBN 978-0240809779

[13] Gartner Website https://www.gartner.com/

[14] Josuttis, N.M. 2007: SOA in Practice: The Art of Distributed System Design

[15] Mark Richards 2015: Microservices vs. Service-Oriented Architecture - ISBN - 978-1491975657

[16] IBM developerWorks article https://www.ibm.com/developerworks/websphere/library/techarticles/1601_clark-trs/1601_clark.html

[17] A Rational Software White paper Alan Brown, Simon Johnston, Kevin Kelly 2002: Using Service-Oriented Architecture and Component-Based Development to Build to Web Service Applications

[18] Brown, P.C. 2011: TIBCO Architecture Fundamentals - ISBN 978-0132762441

[19] Coulouris, George and Dollimore, Jean and Kindberg, Tim and Blair, Gordon 2011: Distributed Systems: Concepts and Design - ISBN 978-0132143011

[20] TIBCO Documentation of BusinessWorks 5, https://docs.tibco.com/products/tibco-activematrix-businessworks-5-13-0

[21] TIBCO ActiveMatrix BusinessWorks 5 - Concepts, https://docs.tibco.com/pub/activematrix_businessworks/5.13.0/doc/pdf/tib_bw_concepts.pdf

[22] TIBCO Documentation of BusinessWorks 6, https://docs.tibco.com/products/tibco-activematrix-businessworks-6-3-5

[23] TIBCO ActiveMatrix BusinessWorks 6 - Application Development, https://docs.tibco.com/pub/activematrix_businessworks/6.4.2/doc/pdf/TIB_BW_6.4.2_application_development.pdf

[24] TIBCO Documentation of EMS, https://docs.tibco.com/products/tibco-enterprise-message-service-8-3-0

[25] TIBCO ActiveMatrix BusinessWorks 6 - Concepts, https://docs.tibco.com/pub/activematrix_businessworks/6.4.2/doc/pdf/TIB_BW_6.4.2_concepts.pdf

[26] Press release, 2014, TIBCO Announces TIBCO ActiveMatrix BusinessWorks 6.0 https://www.tibco.com/press-releases/2014/tibco-announces-tibco-activematrix-businessworks-60-unprecedented-integration

[27] Mulesoft Website, https://www.mulesoft.com/platform/enterprise-integration

[28] Dossot, D. and D'Emic, J. and Romero, V. 2014 Manning: Mule in Action - ISBN 978-1617290824

[29] RAML Website, http://raml.org/

[30] Evans, S. 2017 CreateSpace Independent Publishing Platform: Biztalk: For Starters - ISBN 978-1546918844

[31] Developing SOA Applications with Oracle SOA Suite, https://docs.oracle.com/middleware/1221/soasuite/develop/SOASE.pdf

[32] Modeling and Implementation Guide for Oracle Business Process Management, https://docs.oracle.com/cd/E23943_01/doc.1111/e15176.pdf

[33] Gamblin, R. and Williams, N. and Redbooks, IBM 2017 IBM Redbooks: IBM z Systems Integration Guide for the Hybrid Cloud and API Economy - ISBN 978-0738455907

[34] What is a Framework, www.dsc.ufcg.edu.br/~jacques/cursos/map/html/frame/oque.htm

[35] Codit Integration Framework, https://www.codit.eu/portugal/servicos/integration-framework/

[36] Oracle WebLogic Integration Framework, https://docs.oracle.com/cd/E13214_01/wli/\docs70/aiover/2intfra.htm

[37] TIBCO setEPR Activity, https://docs.tibco.com/pub/activematrix_businessworks/6.3.1/\doc/html/GUID-7E74A88F-1C6F-4DD8-B321-6D90EF9F8F94.html

[38] B. Shim and S. Choue and S. Kim and S. Park, 2008: A Design Quality Model for Service-Oriented Architecture https://ieeexplore.ieee.org/document/4724572/

[39] Eusgeld, I. and Freiling, F. and Reussner, R.H., 2008: Dependability Metrics: GI-Dagstuhl Research Seminar, Dagstuhl Castle, Germany, October 5 - November 1, 2005, Advanced Lectures - ISBN 978-3540689461

A

POC using Framework - SFW

In this appendix is presented an example of the amount of code that a developer needs to implement, using the developed service framework, to develop a full transactional flow.

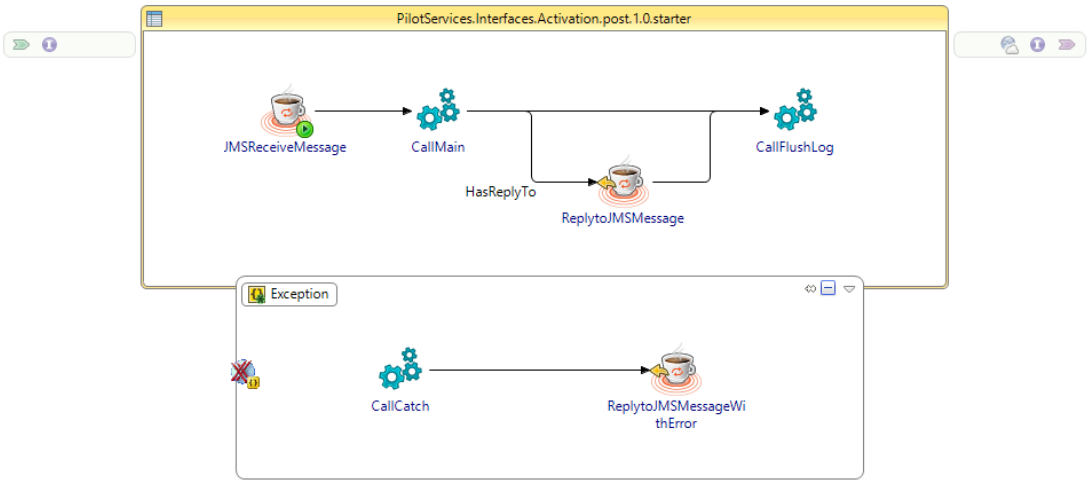


Figure A.1: Interface Starter Example

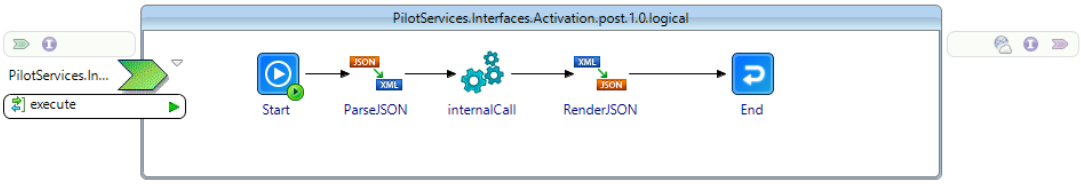


Figure A.2: Interface Logical Example

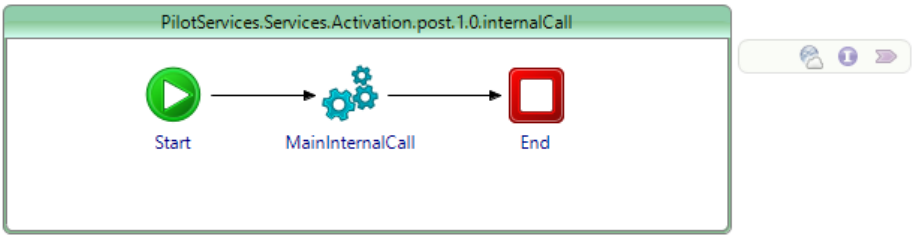


Figure A.3: Internal Call Example

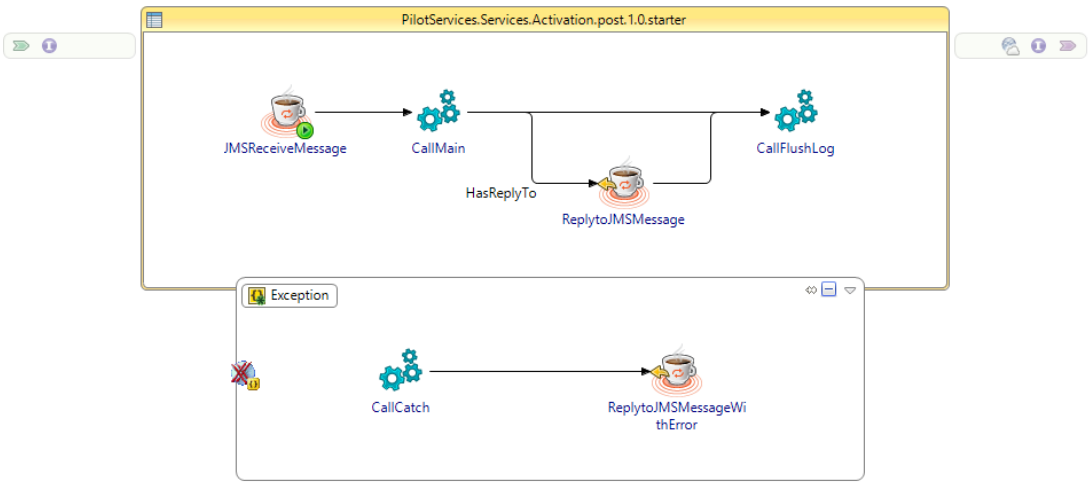


Figure A.4: Service Starter Example

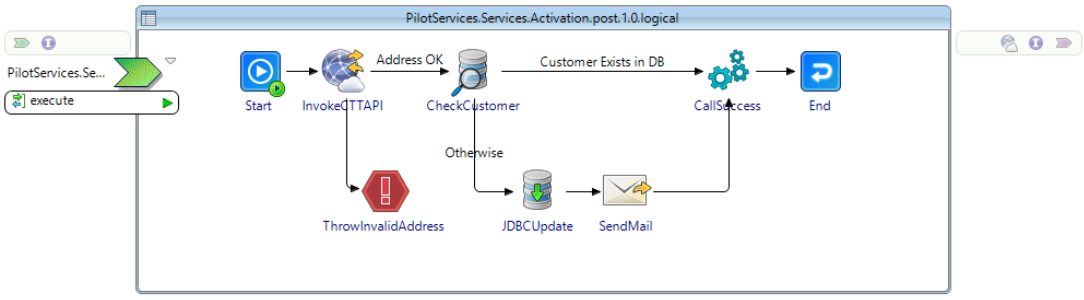


Figure A.5: Service Logical Example

B

POC not using Framework - NSFW

In this appendix is presented an example of the amount of code that a developer needs to implement, on top of the code presented in the previous appendix, to develop a full transactional flow.

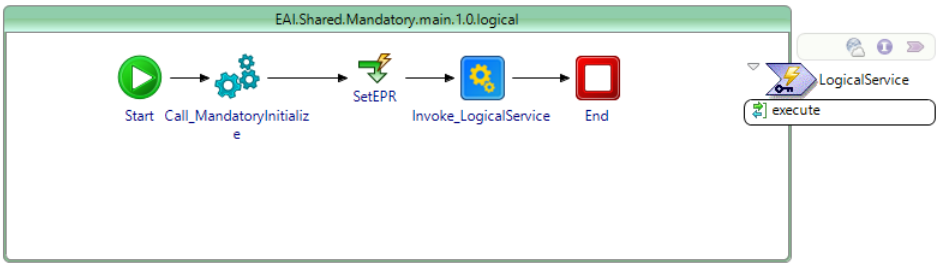


Figure B.1: Main Subprocess

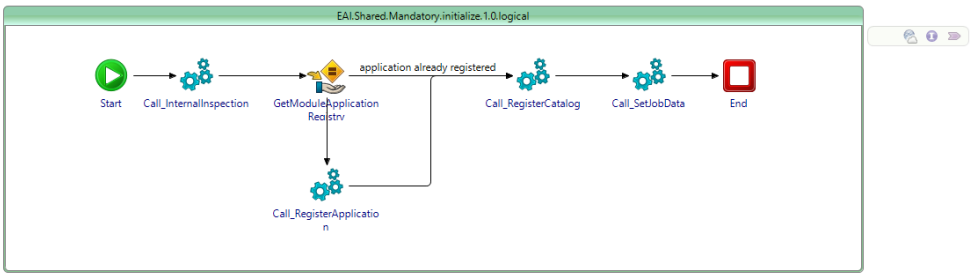


Figure B.2: Initialize Subprocess

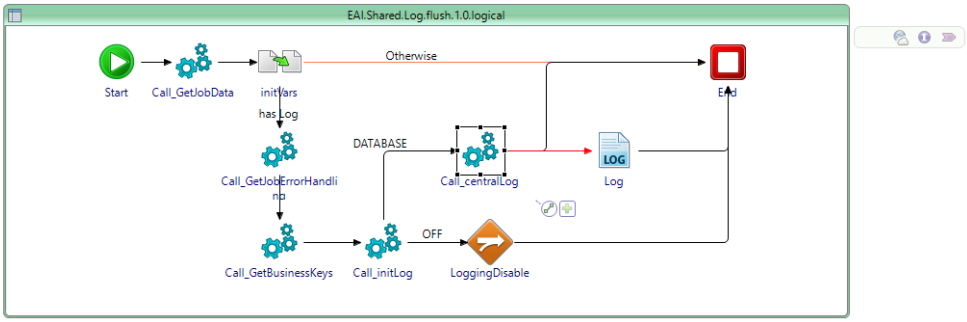


Figure B.3: FlushLog Subprocess

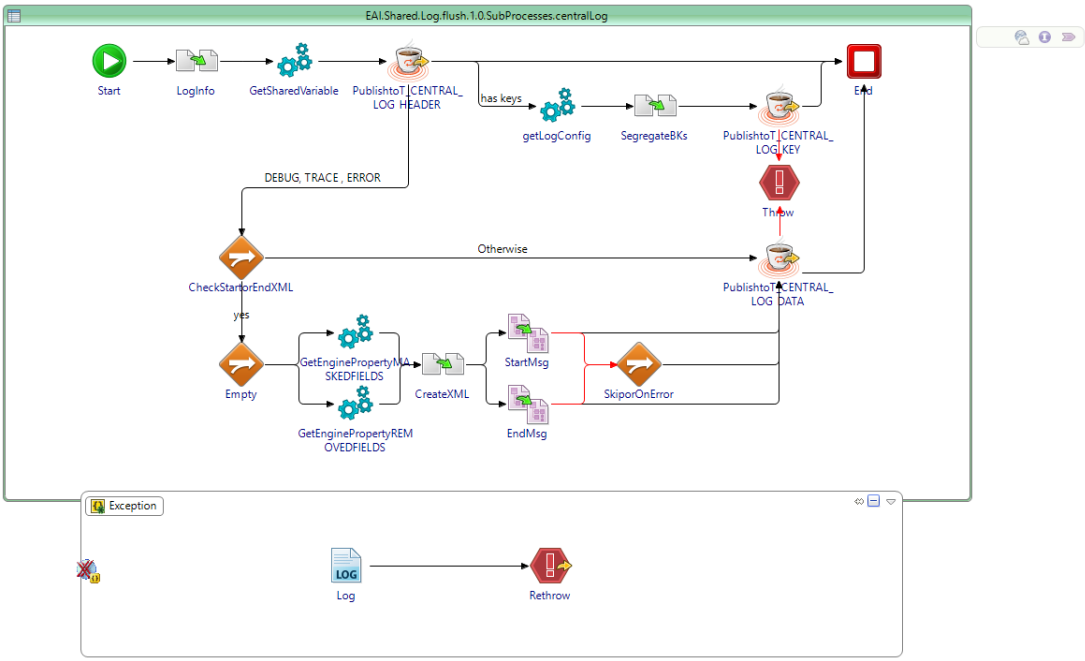


Figure B.4: CentralLog Subprocess

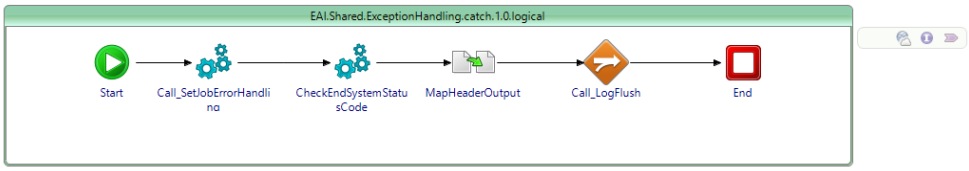


Figure B.5: Catch Subprocess