Service Integration Framework for TIBCO ActiveMatrix® BusinessWorks™ 6

Extended Abstract

Diogo Rebelo da Cruz
Instituto Superior Técnico, Universidade de Lisboa
Lisbon, Portugal
diogo.rebelo.cruz@tecnico.ulisboa.pt

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 strong and powerful 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

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].

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** that implements a set of patterns, guidelines and best practices for the integration developers using TIBCO ActiveMatrix® BusinessWorks™ 6. 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 AND RELATED WORK

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].

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[8].

These services have a set of desired properties[14][15] previously defined. For example, they need to be Visible, Stateless or Reusable.

Additionally to these desired properties, services also need to be **loosely coupled**. Loosely coupled services are connected to other services and clients using standard, dependency-reducing, decoupled message-based methods[15]. When dependencies are minimized, modifications have minimized effects, and the system still runs when parts of it are broken or offline[14].

The language used by TIBCO to implement services and every other function is the Business Process Model Notation.. BPMN is a visual notation to create, share and improve knowledge on business processes[6].

A framework is already defined for the previous version of TIBCO ActiveMatrix® BusinessWorks™.

TIBCO ActiveMatrix® BusinessWorks™ 5 Framework

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. The used adaptation is named **Enterprise Message Service** (Fig. 1), or EMS. It is the TIBCO implementation of Java Message Service which provides queue and topic APIs to send messages[21].

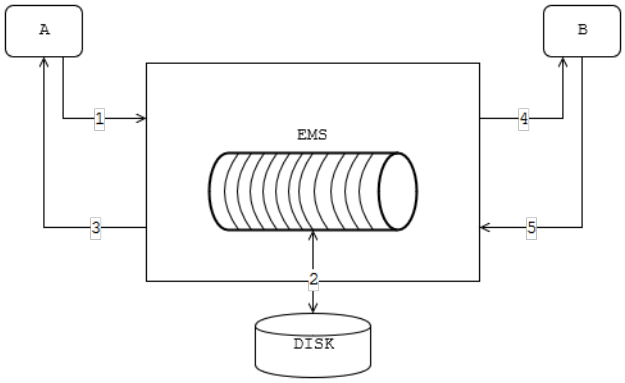

Figure 1: EMS Architecture

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.

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

- **Services**: business functions, defined and implemented by an organization, that are exposed externally, or internally, to the company[2].

- **Processes**: set of activities that interact to achieve a result[5].
- **Interfaces**: 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.

The functions write in the Central Log by calling a process called FlushLog.

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.

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.

3 IMPLEMENTATION

The implementation of the solution went through a series of steps presented in this section.

Architecture

The BusinessWorks™ 5 framework is designed to have a shared module that have all the logic of the framework âĂIJattachedâĂİ 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.2, 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.

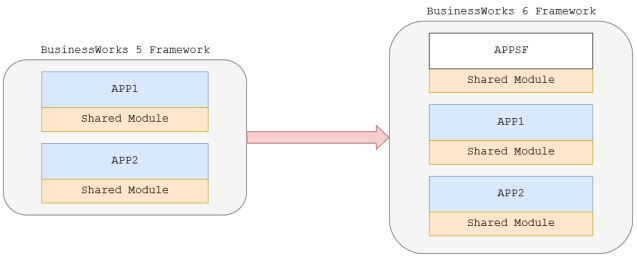

Figure 2: Framework Update

This change was decided in order to maintain the **framework logic transparent** to the developers and to **improve the performance** of the running applications.

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 just receive the request by the EMS, call the Logical, answer to the request and register the transaction in the log. The Logical have the actual logic of the function. Whenever any of this functions returns an error it will be caught like an exception in Java.

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 era of API and Mobile, 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 (SetEPR)[34] that create an endpoint reference in all the processes of the shared module (Fig.3). 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 and starter.

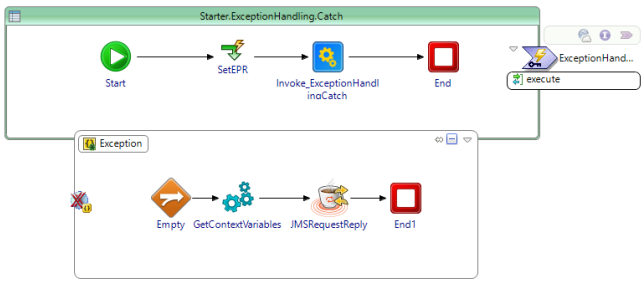

Figure 3: Catch Process

Logging

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

The initial framework has a relational model between the three databases (Header, Keys and Data). This aspect changed to an Analytical Model, or a star schema. 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.

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.

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.

Deployment and Administrations

The administration now will be done in TIBCO Enterprise Administrator. With the evolution of BusinessWorks^{TM}, 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. 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. 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.

4 EXPERIMENTAL EVALUATION

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[35]:Metrics Definition, Metrics Application and Result Analysis.

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[36].

We analyzed the 6 attributes but not all were suited so we defined four based on the six defined in the ISO.

The resulting set of quality attributes are described in Table 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 1: Target 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 2

The identified characteristics are measured using four group of metrics that are defined according to their role and scope[35].

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 2: SOA Design Properties

Service Internal Metrics. Represented in Table 3, use service internal elements like activities, operations, message features and service name.

Metric	Name
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 3: Service Internal Metrics

Service External Metrics. Table 4 are service external metrics and use information from the connected services. It allows to measure the features of consumer and produces services.

System Metrics. Table 5, measure the characteristics of the system in general.

Derived Metrics. The final group makes use of the three previously defined metrics groups. Metrics in this group measure design properties. Metrics that can be derived are:

- DM_ADCS - Average Number of Directly Connected Services : $\frac{SEM\_NDPS + SEM\_NDCS}{SM\_SSNS}$

Metric	Name
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 4: Service External Metrics

Metric	Name
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: System Metrics

- DM_IAUM - Inverse of Average Number of Used Message : $\frac{SM\_SSNS}{SM\_TMU}$
- DM_NO - Number of Operations : $SM\_NSO + SM\_NAO x 1.5$
- DM_NS - Number of Services : SM_SSNS
- DM_AOMR - Squared Avg.Number of Operations to Squared Avg. Number of Messages : $\frac{(\frac{SM\_NAO+SM\_NSO}{SM\_SSNS})^2}{(\frac{SM\_TMU}{SM\_SSNS})^2}$
- DM_CPR - Coarse-Grained Parameter Ratio : $\frac{SM\_NSO + SM\_NAO - SM\_NFPO}{SM\_NSO + SM\_NAO}$
- DM_ANSOR - Adequately Named Service and Operation Ratio : $\frac{SM\_SSNS - SM\_NINS}{SM\_SSNS x 2} + \frac{SM\_NSO + SM\_NAO - SM\_NINO}{(SM\_NSO + SM\_NAO) x 2}$

Table 6 mappes the derived metrics group to 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 7 [35].

Metrics Application

In this chapter is presented all the metrics defined in the previous chapter for two identical **POC - Proof of Concept**

Metric	Design Property
DM_ADCS	Coupling
DM_IAUM	Cohesion
DM_NO	Complexity
DM_NS	Design Size
DM_AOMR	Service Granularity
DM_CPR	Parameter Granularity
DM_ANSOR	Consumability

Table 6: Derived Metrics

Quality Attribute	Description
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 7: Quality Attributes Formulas

- projects. One using the integration framework develop in this work - **SFW (Service Framework)** - and other without any framework - **NSFW (Not Service Framework)**.

The application NSFW is much more complex. Because it didn't use the framework the developer had to implement all the logic and all the resources that the project required

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

Internal Metrics. Table 8 presents the internal metric results obtained in each POC application.

Metric	NSFW	SFW
SIM_NO	27	6
SIM_NFPO	14	0
SIM_NMU	24	12
SIM_NAO	0	0
SIM_NSO	27	6
SIM_NINO	2	2

Table 8: Service Internal Metrics Values

External Metrics. Table 9 presents the external metric results obtained in each POC application.

Metric	NSFW	SFW
SEM_NCSL	7	7
SEM_NDPS	0	0
SEM_NDCS	3	3
SEM_NTPS	1	1
SEM_NTCS	6	6

Table 9: Service External Metrics Values

System Metrics. Table 10 presents the system metric results obtained in each POC application.

Metric	NSFW	SFW
SM_SSNS	3	3
SM_NINS	0	0
SM_NINO	1	1
SM_TMU	24	12
SM_NAO	0	0
SM_NSO	27	6
SM_NFPO	14	0
SM_NPS	21	1
SM_NIS	0	0
SM_NBS	0	0

Table 10: System Metrics Values

Derived Metrics. Table 11 presents the derived metric results obtained in each POC application.

Quality Attributes

Table 12 presents the quality attributes results obtained in each POC application.

Metric	Design Property	NSFW	SFW
DM_ADCS	Coupling	1	1
DM_IAUM	Cohesion	0.125	0.25
DM_NO	Complexity	27	6
DM_NS	Design Size	3	3
DM_AOMR	Service Granularity	1.266	0.25
DM_CPR	Parameter Granularity	0.481	1
DM_ANSOR	Consumability	0.536	0.639

Table 11: Derived Metrics Values

Quality Attribute	NSFW	SFW
Effectiveness	0.907	0.743
Understandability	-19.858	-6.065
Flexibility	0.846	0.543
Reusability	0.463	0.069

Table 12: Quality Attributes Values

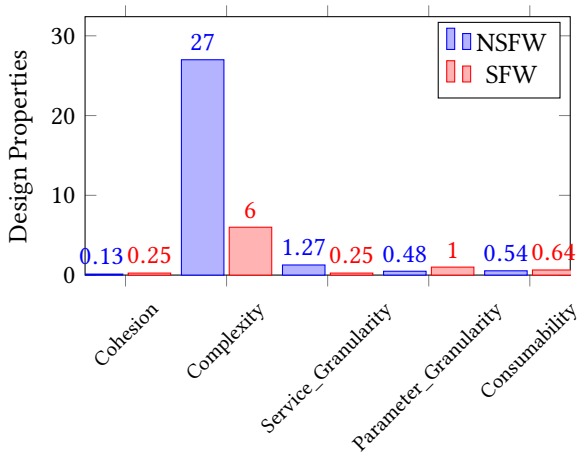

Figure 4: Design Properties Values Comparison

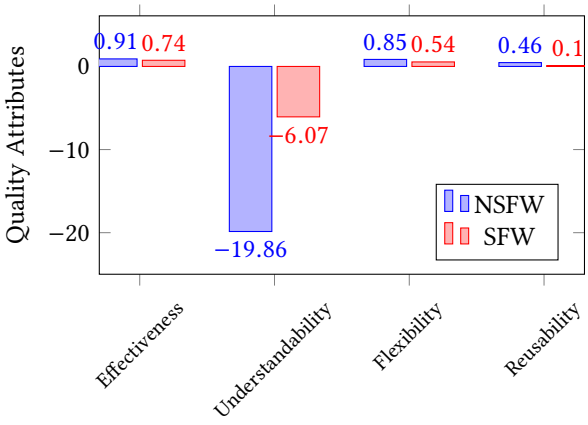

Figure 5: Quality Attributes Values Comparison

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.

These results 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. 4, 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.

Because the values of Coupling and Design Size are the same in both applications they are not represented in the chart, for visibility purposes.

According to the Fig. 5, 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. 4.

The remaining quality attributes presents a better view on the application that did not use the framework. 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.

5 CONCLUSION

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.

REFERENCES

[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] 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

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

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

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

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

[20] 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

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

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

[23] 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

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

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

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

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

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

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

[30] 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

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

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

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

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

[35] 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/

[36] 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