Action Oriented Workflow Framework

A dissertation submitted to the University of Manchester for the degree of Master in Advanced Computer Science in the Faculty of Engineering and Physical Sciences

2016

Blerina Kapllani

School of Computer Science
# Table of Contents

Abstract................................................................................................................................................... 7
Declarations.................................................................................................................................................. 8
Intellectual property statement................................................................................................................... 8
Acknowledgments......................................................................................................................................... 9
The Author................................................................................................................................................... 9
Chapter 1. Introduction............................................................................................................................... 10
  1.1 Project Motivations............................................................................................................................. 10
  1.2 Aim and Objectives................................................................................................................................ 14
  1.3 Report Outline...................................................................................................................................... 15
Chapter 2. Background.................................................................................................................................. 16
  2.1 Chapter Overview.................................................................................................................................. 16
  2.2 Technical Context.................................................................................................................................. 16
    2.2.1 An Introduction to Service Oriented Architectures................................................................. 16
    2.2.2 Web Services.................................................................................................................................. 18
    2.2.3 Comparison of SOAP and RESTful Web Services................................................................. 19
    2.2.4 Enterprise Application Integration and Enterprise Service Bus............................................. 20
    2.2.5 Comparison of Enterprise Service Bus solutions...................................................................... 21
    2.2.6 An introduction to Business Process Management in Enterprise ...................................... 22
    2.2.7 An Introduction to Workflow Based Business Process Management.................................... 24
    2.2.8 Conventional Workflow Models................................................................................................. 25
    2.2.9 State Based Workflow Models................................................................................................. 25
    2.2.10 Petri Net....................................................................................................................................... 26
    2.2.11 Event Condition Action (ECA) state chart.............................................................................. 28
    2.2.12 The need for non workflow BPM methods............................................................................ 28
  2.3 Existing Solutions.................................................................................................................................. 30
    2.3.1 Existing Solutions based on Conventional Workflow model--------------------------------------- 30
    2.3.2 Existing Solutions based on Event Condition Action model.................................................. 31
    2.3.3 Existing Solutions based on Petri Net....................................................................................... 31
    2.3.4 Existing BPM tools.................................................................................................................... 32
    2.3.5 AOW position among existing solutions.................................................................................. 34
  2.4 Summary............................................................................................................................................... 34
Chapter 3. AOW Framework Design........................................................................................................ 36
  3.1 Chapter Overview................................................................................................................................. 36
Chapter 5. Evaluation ..................................................................................................................... 82
  5.1 Chapter Overview ................................................................................................................. 82
  5.2 Evaluation of Workflow Modelling and Execution .............................................................. 82
    5.2.1 Reflection of real world scenarios ................................................................................. 83
    5.2.2 Customer Base Segmentation scenarios ........................................................................ 84
    5.2.3 Loyalty scenario ............................................................................................................ 87
    5.2.4 Analysis on workflow modelling and execution evaluation ......................................... 89
  5.3 AOW framework Testing...................................................................................................... 90
  5.4 Evaluation of Centralisation ............................................................................................... 92
  5.5 Evaluation of Lightweight .................................................................................................. 93
  5.6 Evaluation of Reusability .................................................................................................... 96
    5.6.1 Reusability evaluation of CBS scenarios ...................................................................... 96
    5.6.2 Reusability evaluation of Loyalty Scenario ................................................................. 96
    5.6.3 Conclusions on Reusability Evaluation ........................................................................ 97
  5.7 Summary ............................................................................................................................... 98
Chapter 6. Conclusions and Future Work ..................................................................................... 100
  6.1 Conclusion .......................................................................................................................... 100
  6.2 Future Work ........................................................................................................................ 101
8 References ........................................................................................................................................ 103
Appendix A: Customer Base Segmentation Promotions workflow modelling ........................................ 106
Appendix B: Implementation Details ................................................................................................. 108
  B.1 Database Connection Pooling, Data Source Configuration .................................................. 108
  B.2 Service and Framework Command Execution source code ................................................ 109
  B.3 Example of Triggers translated to a Framework Command ................................................ 109
  B.3 Example of a RESTful Framework Command: .................................................................... 110
  B.4 Framework Command Response: Result Message list ...................................................... 111
  B.5 Web Server Request and Response example ...................................................................... 111
  B.6 File reader configurations ................................................................................................. 113

Final Word Count: 21,934
List of Figures

Figure 1.1 | First Call Event Workflow in Telecom Architecture ......................................................... 11
Figure 1.2 | Balance Top Up in Telecom Architecture ......................................................................... 12
Figure 1.3 | Introduction of Action Oriented Workflow Framework in Telecom Architecture .......... 13
Figure 2.1 | Service Oriented Architecture Concept (Source: Michale Rosen, Boris Lublinsky, Kevin T. Smith, Marc J. Balcer, 2012) ............................................................................................................ 17
Figure 2.2 | Web Service Architecture: a) shows how the web service operates, while b) defines the web service programming stack. (Source: K. Gottschalk, S. Graham, H. Kreger, J. Snell, 2002, Web services actors, objects, and operations, Web services programming stack) ........................................ 18
Figure 2.3 | Business Process Management High Level Architecture (Source: Mathias Waske, 2012, Business Process Management systems architecture model) ........................................................................... 23
Figure 2.4 | Workflow Net Sequential routeing (Source: W.M.P. van der Aalst, 1998, Sequential routeing) ................................................................................................................................................ 27
Figure 2.5 | Workflow Net Parallel routeing (Source: W.M.P. van der Aalst, 1998, Parallel routeing) .............................................................................................................................................................. 27
Figure 2.6 | Workflow Net Conditional routeing (Source: W.M.P. van der Aalst, 1998, Conditional routeing) ................................................................................................................................................ 27
Figure 2.7 | Workflow Net Explicit routeing (Source: W.M.P. van der Aalst, 1998, Explicit choice between B and C based on workflow attribute x.) ................................................................................ 27
Figure 2.8 | System Thinking Theory in BPM (Source: Ilia Bider, Stockholm University, 2012) .......... 29
Figure 2.9 | Oracle BPM Suite Application Architecture (Source: Mark Wilkins, 2010, BPM Conceptual Architecture) ...................................................................................................................... 32
Figure 3.1 | Balance Top Up Event Flow ............................................................................................... 39
Figure 3.2 | Lazarus Scenario Workflow model .................................................................................... 39
Figure 3.3 | Methodology: An Incremental and Iterative approach ...................................................... 40
Figure 3.4 | High Level Architecture of Action Oriented Workflow Framework as a full BPM Suite 41
Figure 3.5 | High Level Architecture of Action Oriented Workflow Framework as a simplified BPM Engine .................................................................................................................................................. 42
Figure 3.6 | AOW Data Model ............................................................................................................... 46
Figure 3.7 | AOW Data Model: Components Model ............................................................................ 47
Figure 3.8 | Next Action Algorithm Design .......................................................................................... 51
Figure 3.9 | Workflow Initiation phase .................................................................................................. 53
Figure 3.10 | Workflow Engine control flow .......................................................................................... 54
Figure 3.11 | BlockingQueue producer and consumer ........................................................................... 55
Figure 3.12 | Flow Executor Manager, Consume Flow .......................................................................... 56
Figure 3.13 | Flow Executor, workflow execution process ...................................................................... 57
Figure 4.1 | Execute Action Method ..................................................................................................... 64
Figure 4.2 | Workflow Initiation phase: Flow Service execution process ............................................ 66
Figure 4.3 | Workflow Initiation Phase, Workflow Execution Phase .................................................... 68
Figure 4.4 | Framework Command Executor: Execute Command ................................................... 69
Figure 4.5 | File Reader Workflow ....................................................................................................... 76
Figure 5.1 | Adaption of AOW in a mirror of Telecom Environment ...................................................... 83
Figure 5.2 | Loyalty Program, Point Rewarding Workflows ................................................................. 87
Figure 5.3 | Loyalty Program, Point Redemption Workflow .................................................................... 88
Figure 5.4 | Run Result of Unit Test Suite ............................................................................................ 90
Figure 5.5 | AOW Framework Code Coverage report ........................................................................... 91
Figure 5.6 | White Box Testing, Pick Flow test case scenario example .................................................. 92
Figure A.1 | Customer Base Segmentation Promotion Workflow ............................................................ 106
Figure A.2 | Components Model, Flow Configurations.............................................................................. 107
List of Tables

Table 1.1 | Provisioning Workflow event – action series ................................................................. 11
Table 1.2 | Balance Top up Workflow event – action series ........................................................... 12
Table 2.1 | SOAP and Restful Web Service Characteristics ............................................................ 19
Table 2.2 | Comparison of ESB solutions ....................................................................................... 22
Table 2.3 | Advantages and disadvantages of Oracle BPM Suite ..................................................... 33
Table 2.4 | IFTTT advantages and disadvantages as BPM tool ......................................................... 33
Table 2.5 | Positioning of AOW among existing solutions. a) Categories of workflow models. b) Capabilities of existing solutions .......................................................................................................... 34
Table 2.6 | BPM approaches characteristics ..................................................................................... 35
Table 3.1 | AOW Framework Terms and Definition ........................................................................... 37
Table 3.2 | Lazarus Promotion Scenario translated into tabular form ................................................ 38
Table 3.3 | Construction Scenario, Hybrid Framework Command Configurations ................................ 44
Table 3.4 | Construction Scenario, Flow Configurations ..................................................................... 47
Table 3.5 | Construction Scenario, Get Segment Action Configurations ........................................... 48
Table 3.6 | Construction Scenario, Action Framework Command Configurations ................................ 49
Table 3.7 | Construction Scenario, Pick Flow Rule Configuration .................................................... 49
Table 3.8 | Construction Scenario, Next Action Rules Configuration .............................................. 52
Table 3.9 | Queues capabilities and features ..................................................................................... 55
Table 3.10 | Advantages of Thread Pool over One Request - One Thread ........................................ 57
Table 3.11 | Most Common database operations and related frequencies ........................................ 59
Table 4.1 | The technologies used to implement AOW Framework .................................................... 62
Table 4.2 | Service Object Configuration .......................................................................................... 65
Table 4.3 | RESTful web Client configurations ................................................................................. 67
Table 4.4 | Framework Command Types .......................................................................................... 69
Table 4.5 | Triggering Framework Command Adapters ......................................................................... 70
Table 4.6 | Action Framework Command Parameters Categorised by Service ................................... 71
Table 4.7 | Action Framework Command Adapters: extraction of Framework Command Attributes from Service Object responses ........................................................................................................... 73
Table 4.8 | REST Response Parameter Configurations ......................................................................... 74
Table 4.9 | File Reader Workflow Actions .......................................................................................... 77
Table 4.10 | File Reader Next Action Rules ......................................................................................... 77
Table 5.1 | Reflection of Telecommunication Environment. Systems that has been implemented to support the scenarios simulation ............................................................................................................. 84
Table 5.2 | Evaluation Scenarios Definition. Five Marketing Requests Specifications in Cluster Base Segmentation Promotions ............................................................................................................ 85
Table 5.3 | Five Customer Base Segmentation Promotions used as evaluation scenarios .................... 85
Table 5.4 | Components Model Configuration Summery ....................................................................... 86
Table 5.5 | Loyalty Program, Point Rewording and Birthday workflow Components Model Configuration Synthesis ................................................................................................................................. 88
Table 5.6 | Loyalty Program, Point Redemption workflow Components Model Configuration Synthesis ................................................................................................................................. 89
Table 5.7 | AOW architecture. A composition of lightweight technologies ........................................ 93
Abstract

Business Process Management (BPM) is the activity that improves enterprise performance by coordinating business processes and operations with its resources including technology and people. Many BPM solutions in the market operate on a Service Oriented Architecture. However, some of these solutions are heavyweight middleware systems supported by a complicated architecture. On the other side, there are lightweight solutions that do not provide reusability or are not a fit for an enterprise environment.

This project aims in piloting a centralised, lightweight framework that enables the business process implementation in the enterprise using reusable components. It operates in a Service Oriented Architecture. This BPM tool adapts ECA state based workflow model to benefit from the flexibility that this model provides. The design is composed of two main components, Design Time that defines the Data Model and Run Time that runs the workflows using the Data Model. The integration of this tool with other systems is performed using the RESTful Web Services and Talend Enterprise Service Bus.

The project is evaluated against the aim and objectives by the simulation of six evaluation scenarios in a reflection of real Telecommunication Environment. They point out the ability of the framework to model and execute business processes in this enterprise domain. The pilot is an orchestration of lightweight technologies. However, usability needs to be improved by adding a graphical component that will visualise and provide guidelines for workflow modelling as well as system controls that will prevent configuration errors. This project is promising regarding reusability. The design enables reusability in service level and action level. When patterns are identified in the business processes of subdomains, then the reusability might reach up to 100%. However, additional functionalities need to be introduced to enforce the benefits of this system capability.
Declaration

No portion of the work referred to in the dissertation has been submitted in support of an application for another degree or qualification of this or any other university or other institute of learning.

Intellectual property statement

i. The author of this dissertation (including any appendices and/or schedules to this dissertation) owns certain copyright or related rights in it (the “Copyright”) and s/he has given The University of Manchester certain rights to use such Copyright, including for administrative purposes.

ii. Copies of this dissertation, either in full or in extracts and whether in hard or electronic copy, may be made only in accordance with the Copyright, Designs and Patents Act 1988 (as amended) and regulations issued under it or, where appropriate, in accordance with licensing agreements which the University has entered into. This page must form part of any such copies made.

iii. The ownership of certain Copyright, patents, designs, trademarks and other intellectual property (the “Intellectual Property”) and any reproductions of copyright works in the dissertation, for example graphs and tables (“Reproductions”), which may be described in this dissertation, may not be owned by the author and may be owned by third parties. Such Intellectual Property and Reproductions cannot and must not be made available for use without the prior written permission of the owner(s) of the relevant Intellectual Property and/or Reproductions.

iv. Further information on the conditions under which disclosure, publication and commercialisation of this dissertation, the Copyright and any Intellectual Property and/or Reproductions described in it may take place is available in the University IP Policy, in any relevant Dissertation restriction declarations deposited in the University Library, and The University Library’s regulations.
Acknowledgments

I would like to express my sincere gratitude to my supervisor Prof. Carole Goble for being my mentor during this project. I thank her for giving me the support, advice and guidance to deliver this project and this dissertation.

I would also like to express my thanks to Stian Soiland-Reyes for the technical advice and guidance during the system design and implementation.

The Author

I graduated in Computer Science from the University of Tirana, Albania on 2008 and I finished my first MSc in Computer Science at the same university on 2010.

I have in total seven years working experience as a Software Engineer and Technical Architect in different industries like Education, Telecommunication and Start Ups. For the last five years, I was working for Vodafone dealing with online charging systems. During this period I designed online charging services in an event driven distributed environment. I have considerable experience with SOA architectures, system integration, and middleware.

This project idea has started when I was working at Vodafone. I experienced the need for a lightweight BPM tool that offered reusability in that dynamic environment. The existing solutions in Oracle were robust and with high performance, but the business processes development was time-consuming and challenging, while marketing was pushing for more efficient deliveries and better time to market.
Chapter 1. Introduction

This project focuses on the creation of a *lightweight* Action Oriented Workflow (AOW) framework for Business Process Management, which is a middleware that aims to serve as workflow based BPM tool for the enterprise. It works as a state machine where the workflows are collections of possible actions. The flow of these actions is defined in runtime by a set of rules applied over the state transitions. The business processes and services will be translated in these workflows using configurations. It will facilitate the business process implementation using *reusable components*. AOW will operate as a middleware in a Service Oriented Architecture [1]. Therefore, it provides a communication layer that enables system integration.

There is not a clear and full definition of Business Process Management (BPM) as a discipline, even though its applicability is acknowledged. Based on analysis on how BPM is applied in the enterprise, it can be seen as a combination of techniques, methods, and tools that offer automation, modelling, monitoring, and optimisation of the business processes by coordinating the operations of a company with its resources (technology and people) [2] [3]. The efficiency of this coordination is measured directly by business success. The goal of constantly improving the business process management within an organisation is to facilitate and automate the business operations, to offer new services and products with reduced cost, and to improve time to market [2].

This section serves as an introduction to this project, detailing motivations, aim, and objectives.

1.1 Project Motivations

AOW operates as a state machine using Event Condition Action (ECA) state based workflow model that will be explained in section 2.2.911. This model is useful in dynamic and event driven enterprise environments such as Telecommunication. Based on the previous experience of the author in telecommunication domain, it is observed that the business processes in this environment can be intuitively mapped into an Event Condition Action workflow model.

Two real world scenarios are provided below to point out the need for a BPM framework that offers a mechanism to orchestrate *Event - Action* processes demonstrated below.
Before going in scenario details, a terminology list is provided:

- Home Location Register is the central database of the core network maintaining identification and other valuable customer information [4].
- Billing [6] is responsible for charging the traffic that the customer performs.
- SMS Centre receives and delivers SMS [4].

**Scenario 1:** Figure 1.1 demonstrates a customer provisioning workflow that is triggered by the First Call event. This event is generated by the customer when he performs the first call with a new mobile number.

![Figure 1.1: First Call Event Workflow in Telecom Architecture](image)

The customer provisioning workflow is represented by the set of $Event \Rightarrow Action$ listed below:

<table>
<thead>
<tr>
<th>Event (generated on)</th>
<th>Action (executed on)</th>
</tr>
</thead>
<tbody>
<tr>
<td>First Call (Mobile)</td>
<td>Update Customer Status (Home Location Register)</td>
</tr>
<tr>
<td>Status Change (Home Location Register)</td>
<td>Check Promotion Eligibility (Customer Relationship Management)</td>
</tr>
<tr>
<td>Eligible (Customer Relationship Management)</td>
<td>Subscribe Bonus (Billing)</td>
</tr>
<tr>
<td>Bonus Subscription (Billing)</td>
<td>Send notification SMS (SMS Centre)</td>
</tr>
</tbody>
</table>

**Table 1.1**: Provisioning Workflow event – action series

A process should be implemented to model and execute the set of $Event \Rightarrow Action$ illustrated in Table 1.1.
Scenario 2: Figure 1.2 demonstrates a promotion campaign triggered by the event of Balance Top Up.

Balance Top Up workflow is represented by the set of Event => Actions listed below:

<table>
<thead>
<tr>
<th>Event (generated on)</th>
<th>Action (executed on)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Balance Top UP (Electronic Top Up Application)</td>
<td>Top Up Balance (Billing)</td>
</tr>
<tr>
<td>Balance Increased (Billing)</td>
<td>Check Promotion Eligibility (Customer Relationship Management)</td>
</tr>
<tr>
<td>Eligible (Customer Relationship Management)</td>
<td>Subscribe Bonus (Billing)</td>
</tr>
<tr>
<td>Bonus Subscription (Billing)</td>
<td>Send notification SMS (SMS Centre)</td>
</tr>
</tbody>
</table>

Another process should be implemented to model and execute the series of Event => Actions illustrated in Table 1.2.

AOW is a BPM framework that operates as a state machine using ECA state based model. It can be observed that these two scenarios are naturally translated in Event => Actions. Thus, AOW is a fit for this event driven environment.

The motivations of this project are based on three important aspects that these scenarios point out (Centralisation, Lightweight, and Reusability):

I. Centralisation

Suppose that for each scenario an individual application is developed. The system integration will be performed individually. Taking into consideration that a telecommunication company might have thousands of such scenarios, this approach seems inefficient. A centralised framework is a better option as:

- Integrations will be performed only once in a centralised architecture.
- All workflows of business processes will be consolidated into one system.

This approach simplifies system integration and process management. Therefore, these scenarios illustrate that event - action processes need to be orchestrated by a centralised framework.
Figure 1.3 shows the overall architecture when the centralised solution (Action Oriented Workflow framework) is integrated into it. By introducing this framework, we bring together all scenarios into one system.

![Diagram](image)

Figure 1.3 | Introduction of Action Oriented Workflow Framework in Telecom Architecture

II. Lightweight

There are many existing BPM solutions in the enterprise. Some of these solutions are heavyweight in terms that:

- They are supported by a sophisticated and expensive infrastructure.
- They have complicated application architecture.
- Most of these solutions introduce restrictions and dependencies on technology choices.
- They offer many capabilities, more than what a company might need. This results in tools that are difficult to be used. Their users will need extensive training to benefit from fancy system features.

For example, Oracle PBM solution is compound by the Oracle SOA component. There is an Oracle adapter for a specific integration, and the price varies by the number of the integrations that the customer wants to set up. In addition, for better transparency in the transaction management, the Enterprise Service Bus should be included in the BPM suite. All these applications should be deployed in a Web Server. Only Oracle products have full compatibility on this architecture [7]
On the other side, there are extremely lightweight triggering systems which are limited in functionality or are not a fit for the enterprise environment. Even though they are event driven and dynamic, they support only simple workflows and do not offer a full BPM solution. For example, IFTTT provides capabilities of business process modelling using workflows. It is a lightweight solution but with limited capacities.

Therefore, the need for a lightweight solution is one of the strongest motivations for this project.

**III. Reusability**

On the above telecommunication scenarios are identified two levels of reusability:

- **Reuse of Services.** The service that updates the customer profile on Billing is used in both scenarios. In the first one it updates the promotions, and in the second it updates the balance.
- **Reuse of Actions.** The actions are specific operations of the business process. These might be considered as the most beneficial components regarding reusability. For example, both above scenarios will reuse the same actions to send the notification SMS or to subscribe the bonus to the customer profile.

Most of BPM solutions offer Service reusability, but they lack the Action one. They do not provide the option to save each step of a workflow that is being modelled as an individual component with the perspective of reusing it.

**1.2 Aim and Objectives**

The aim of this project is to design and pilot a centralised, lightweight Action Oriented Workflow framework that will provide reusable components for workflow modelling. It will be integrated into a Service Oriented Architecture using off the shelf technologies.

Objectives of this project are listed below:

- Design a lightweight architecture of a rule based workflow framework using Event Condition Action workflow models.
- Integrate the framework into a Service Oriented Architecture using Web Services.
- Perform an empirical evaluation of the architecture through exercising the pilot in campaign management scenarios that will emphasise two levels of reusability.
Lightweight in this context means:

- The system will not require sophisticated application architecture. It will be a composition of lightweight technologies.
- It will not be restrictive in environment technology choices. Indeed, this application fully supports and encourages open source technologies.
- It will offer simple capabilities for business process modelling. The end user should be able to create the workflows by performing some simple configurations or scripts.

Reusability will be defined in Service and Action levels.

The users of this system are the ones that will model and monitor the business processes implemented using workflows. They should be domain experts with technical knowledge.

1.3 Report Outline

Chapter 2 sets the background of the project by analysing all off the shelf technologies that are used to build the AOW framework. It covers:

- An overview of Service Oriented Architectures.
- Web Services and Enterprise Service Bus solutions.
- High level architecture of Business Process Management in the enterprise in which AOW architecture is based.
- Conventional Workflow and State Based Workflow models.
- Existing solutions in BPM and Workflow Management Systems.

Chapter 3 presents the high level architecture and design, pointing out the challenges and justifying the design choices.

Chapter 4 provides the implementation details of the system by detailing the design provided in chapter 3.

Chapter 5 displays the scenarios simulation in a reflection of real world environment to evaluate the system regarding Centralisation, Lightweight and Reusability.

Chapter 6 concludes and consolidates the results as well as point out improvements and future work.
Chapter 2. Background

2.1 Chapter Overview

In this chapter is described the technical context of this project along with existing solutions in BPM. The technical context provides a good understanding of all off the shelf technologies that this project will use to reach the aim and its objectives. First, an introduction to Service Oriented Architectures and Enterprise Application Integration is provided. In this context, Web Service and Enterprise Service Bus technologies are discussed. A high level architecture of BPM in the enterprise is illustrated. This architecture serves as a guide for the AOW architecture. Different BPM approaches are displayed falling into two broad categories: Workflow based and non-Workflow based. Workflow based BPM models are detailed further in Conventional Workflow Models and State Based Workflow Models. The strengths and weaknesses of each model are analysed. The discussion on existing solutions provides analysis of systems capabilities as well as gives an understating where AOW is positioned among them.

2.2 Technical Context

In this sections are discussed the Service Oriented Architectures and Enterprise Application Integration platforms. Workflow based and non Workflow BPM models are analysed. Workflow based approach is detailed further in conventional and state based workflow models. State based workflow is an important model that this project uses to provide a non-traditional workflow BPM solution using State Charts. Petri Net is provided as an alternative option of state based workflow models.

2.2.1 An Introduction to Service Oriented Architectures

One of the first successful attempts to integrate different systems in a distributed environment has been performed by Banks on 1994. Their approach was exposing distributed objects of their main business logic entities as interfaces that were accessible by other systems [8].

There are many definitions of Service Oriented Architectures (SOA). Usually, it is referred as an architectural paradigm focused on exposing the business logic modules as services. In fact, SOA is a vast concept far more than just another architectural model for enterprise [9]. Figure 2.1 illustrates what successful SOA means.
Architectural Approach: it suggests designing the business logic in decoupled modules that offers different services [8].

Business Vision: It implies a strategy on how technology can better cope with the agility of the enterprise, by providing services in a flexible, scalable and reusable way, with reduced cost [8] [10]. Agility in these terms means the ability of an enterprise to be adapted with frequent unpredictable changes.

Methodology: It provides methodology on how to perform a design that models business logic as loosely coupled components and exposes them as accessible services.

Reuse Initiative: Reuse of resources is one of the aims of SOA. The client applications call the exposed services to be executed remotely, without using their resources to do so.

Organisational Structure: A successful SOA implies a centralised management of services that control and root service request to the proper provider.

Technology Mastery and Service Abstraction: this is what we can get from Web Services.

Based on this SOA definition we can conclude in three important features of this architecture:

- Centralisation
- Reusability
- Loose coupling (minimization of dependencies).
Knowing that Centralisation and Reusability are in the focus of this project, SOA is an architectural approach that totally fits in.

### 2.2.2 Web Services

Web Service is an application component that enables the exposure of services over the network in a standardised, layered architectural model [11]. This is one of the most used SOA components that enable the execution of services remotely.

**Web Service Architecture:** Figure 2.2 illustrates the components and layers of the Web Services. Part a) of this figure demonstrates the components of Web Services [11] [12]:

- **Service Provider** is the owner of the Web Service, where the business logic is implemented. It is responsible for exposing the service description in a standard format and publishing it to Service Register (PUBLISH operation).
- **Service Register** serves as a repository of web service descriptions published by their Service Providers.
- **Service Requester** is a client application (commonly a browser) that searches (FIND operation) the Service Repository to use a published Web Service. The requester needs to use the Bind operation to access the service. This operation performs interaction, invocation, and initialization by using the information of service description.

![Web Service Architecture](image)

**Figure 2.2 | Web Service Architecture:** a) shows how the web service operates, while b) defines the web service programming stack. (Source: K. Gottschalk, S. Graham, H. Kreger, J. Snell, 2002, Web services actors, objects, and operations, Web services programming stack).
Part b) of this figure defines the conceptual layers of a Web Service [12]. Below is a description of these layers [11][12]:

- **Network**: This is the lower layer, over which the communication between Service Provider and Service Requester is established. It is represented by network protocols like HyperText Transfer Protocol (HTTP: protocol of the World Wide Web), FTP (File Transferring Protocol), Email protocol, and SMS protocol. This layer manages security and service quality.

- **Message Exchange Layer**: This layer operates in a distributed environment like Simple Object Access Protocol (SOAP).

- **Service Description Layer**: This layer provides the service description information using a standardised mechanism called Web Service Description Language (WSDL).

- **Service Discovery and Service Publications**: These layers enable Service Requesters to search for Web Service Descriptions and Service Providers to publish it.

- **Web Service Flow Layer (WSFL)**: This layer is used to orchestrate Web Services in workflows.

### 2.2.3 Comparison of SOAP and RESTful Web Services

SOAP is already mentioned as a message exchange protocol in distributed environment, having so SOAP-based web services. However, this is not the only type of web service used in Service Oriented Architectures. Representational State Transfer (REST) is another alternative.

Below is a table that defines the characteristics of these two technologies.

<table>
<thead>
<tr>
<th></th>
<th>Network</th>
<th>Service Description Standard</th>
<th>Message Format Standard</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>SOAP</strong></td>
<td>HTTP, FTP etc.</td>
<td>WSDL</td>
<td>Extensible Markup Language (XML)</td>
</tr>
<tr>
<td><strong>REST</strong></td>
<td>HTTP only</td>
<td>Web Application Definition Language (WADL)</td>
<td>XML/ JavaScript Object Notation (JSON)</td>
</tr>
</tbody>
</table>

Table 2.1 | SOAP and Restful Web Service Characteristics

SOAP is considered to be a heavyweight protocol regarding processing and the message size. SOAP keeps track of states during its operations. Moreover, the serialisation of data types to XML is considered a heavyweight process that might issue performance concerns. It seems that REST is a better option regarding *lightweight*. The request for resources is
straightforward, and no track is maintained [13] [14]. REST based web services are usually referred as RESTful Web Services.

RESTful Web Services will be used in this project to enable the communication of the AOW framework with other Information Systems. It will be discussed in more details in Chapter 3 and 4.

2.2.4 Enterprise Application Integration and Enterprise Service Bus

Enterprise Application Integration (EAI) is one of the main concerns of the enterprise environment. Nowadays most of the companies have modularised services provided and managed by different applications. Obviously, these applications need to communicate with each other to complete business tasks [15]. An example of this architecture has been already described in chapter 1 (telecommunication scenarios). System integration directly impacts performance, flexibility, robustness, scalability and maintainability. Following the discussion in section 1.1, below is analysed why Point to Point Integration is not a good architectural approach.

Point to Point Integration is the first integration model. This model introduces many issues, especially with scalability and maintainability. This integration uses interfaces that directly link the systems with each other. If this lightweight architecture integrates up to three systems, then it might work well with benefits in performance. Maintainability is not yet an issue. The problems start when there is a need to add more systems in this integration scheme. As the number of integrations increases significantly, maintainability and scalability become a serious problem [15]. Therefore, the need for a centralised integration node emerges as the number of applications increases.

Centralised Integration: Enterprise Service Bus is one of the most used integration models in Service Oriented Architecture. This component performs message queuing, prioritisation, transformation, and routeing over a distributed environment [15]. Conceptually it is a centralised integration model, but physically this system is replicated in cluster nodes. Automatic failover capability ensures that if one node fails all the requests are sent to the nodes that are up and running. In addition, the transactions are distributed over different instances of this application based on their actual load, improving the performance significantly. This is a highly scalable architecture and one of the best options in system integration for the enterprise.
2.2.5 Comparison of Enterprise Service Bus solutions

There are many ESB solutions in the market. Among them, there are lightweight open source options that can easily be integrated and tailored in a specific environment. In Table 2.2, it is illustrated a comparison of different ESB solutions falling in two categories:

- Commercial ESBs (Oracle ESB, IBM ESB, Microsoft ESB)
- Open source ESBs (Mule, Talend)

Oracle ESB [16], IBM ESB (WebSphere ESB) [17] and Microsoft ESB [18] are part of a SOA suite offering not only the Enterprise Service Bus capability but also Web Service Container (Web Logic, WebSphere, .Net respectively), Business Process Management, Business Activity Monitoring, and SOA security.

Mule ESB [19] is one of the most stable Open Source ESBs which is not part of an SOA suite. Its main capabilities are:

- Service Creation and Hosting: ESB is used as a lightweight Service Container
- Data Transformation: transforms the data from one format or protocol to another.
- Message Routing supports rule based message routing and filtering.
- Service Mediation performs message aggregation and transformation
- Authentication and Security offer service authentication and message encryption.

Talend ESB [20] is another well known open source integration technology that is gaining importance in the enterprise environment. It is part of Talend SOA suite. Besides system integration capabilities, it offers data integration and Big Data manipulation. Talend is based on Apache technologies and is built in Java. It offers flexibility by the use of Java code generator, metadata editor and configuration management.

Commercial options are more stable solutions than open source. They offer almost every possible feature with the high quality of service and security, but they are heavyweight and too complex to get started.
<table>
<thead>
<tr>
<th>Lightweight</th>
<th>Commercial ESB</th>
<th>Open Source ESB</th>
</tr>
</thead>
<tbody>
<tr>
<td>Heavyweight: high resource requirements on run time</td>
<td>Mule</td>
<td>Talend</td>
</tr>
<tr>
<td>Features and functionalities</td>
<td>Very rich in features and functionalities, but many applications use only part of them</td>
<td>Satisfactory features</td>
</tr>
<tr>
<td>Stability</td>
<td>Stable</td>
<td>Satisfactory</td>
</tr>
<tr>
<td>Complexity</td>
<td>Complex architecture, complex solution</td>
<td>Simple architecture, simple solution</td>
</tr>
<tr>
<td>Ease of Use</td>
<td>Complex installation. Need Extensive training to use it</td>
<td>One click installation, easy to get started, intuitive design</td>
</tr>
<tr>
<td>Flexibility</td>
<td>Costs and requires times to modify functionality</td>
<td>Highly extensible</td>
</tr>
<tr>
<td>Community</td>
<td>No community</td>
<td>Limited</td>
</tr>
<tr>
<td>Availability of Source Code</td>
<td>not visible</td>
<td>Restrictive</td>
</tr>
</tbody>
</table>

Table 2.2 | Comparison of ESB solutions

*Lightweight* is one of the most important goals of this project. Obviously, Commercial ESBs are not an option. Both Talend and Mule are lightweight solutions, but Talend seems to have advantages regarding community, availability of source code and features. In addition, Talend jobs are deployed in Apace Tomcat that is one of the most used open source, lightweight Web Servers. Thus, Talend is the best match for this project.

2.2.6 An introduction to Business Process Management in Enterprise

BPM solutions offer options for modelling, monitoring and improving the company processes. The business processes of organisations are domain – specific but this does not infer that they are static. Indeed, these processes are constantly changing along with the technology, and market requirements [21]. Consequently, one of the challenges of the BPM solution is being at the same time domain-specific and generic. It should be flexible to changes, and highly adaptable to the dynamic environment of an enterprise.
Figure 2.3 illustrates a common high level architectural model of a BPM solution. It suggests the modules that such a solution might have and their interactions [2].

Each Business process Management tool has two phases:

- **Design Time**: The business processes are modelled into a language defined by the tool usually into workflows.
- **Run Time**: the modelled business processes are executed using the Process Engine of the tool.

The main modules of this architecture are:

- Business Process Environment generates the triggers for initiating the business process models.
- Business Process Modelling offers the environment to create the models of business processes operations and structures.
- Business Process Model Repository serves as a repository for the business models created by the first component. It provides mechanisms to store the components and to expose them for reuse.
- Process Engine offers the environment to build the processes as compositions of business process models. It also serves as a runtime environment for these compositions.
The processes assembled in Process Engine generate requests for remote execution of services/operations. This indicates that BPM in Enterprise operates in an SOA.

The existing BMP tools implement Workflow based BPM models where the business processes are translated into workflows. However, the need for extensive agility encourages researches for non-workflow BPM models. Both these models will be analysed on the next sections.

2.2.7 An Introduction to Workflow Based Business Process Management

Actually, BPM solutions are workflow based. They model the business processes as a set of operations. BPM usually is confused with Workflow Management Systems but these are two separate concepts, even though in many existing BPM solutions they are tightly used together.

Workflow Management System (WMS) is a technology that enables the creation, and management of workflows, running in one or more workflow engines. BPM is a wider concept (referring to section 2.2.6) [3]. The Workflow itself is a set of operations executed in sequence or parallel. BMP tools make use of WMS to map out the business processes as workflows.

The high level architecture in Figure 2.3 introduces the Process Engine. In the case of Workflow Based BPM tools, this engine is represented by a WMS. It has at least two main sub-components [22]:

- A Workflow Modeller offers the environment to model the workflows. Usually, it is a graphical application that offers scripting and drag and drop facility.
- A Workflow Engine is the execution platform of the workflow. Its main responsibilities are to execute the workflows, request services from service providers and perform monitoring and debugging.

Many workflow models are conventional or state-based models. All these notions will be analysed in the following sections.
2.2.8 Conventional Workflow Models

Conventional BPM solutions make use of WMS to model the workflow as a set of activities organised in a specific predefined order. It improves resource usage by standardising and automating the business processes. [23]

Important properties of the conventional workflow-based BPM modelling are:

- The workflow is comprised of identifiable activities.
- Each activity has predefined inputs and outputs.
- The input/output dependencies between activities are explicitly defined.
- Loops or parallel execution of activities are allowed except cases when the activities have input/output dependencies.
- Each activity is executed by using a specific resource. It ensures the management of resource usage.

Activities are the operations or steps of the business process that is being modelled.

This workflow model is very popular. There are many outstanding WMS and BPM tools that follow this approach. Some of these existing solutions will be analysed in section 2.3. This model offers automation, standardisation, and resource management. The main reason why this workflow model is widely used is the fact that it is simple. Simplicity concerns both the system design and business process modelling.

On the other hand, this workflow model does not provide the required level of flexibility to meet the demand of the enterprise for changes. For environments that experience frequent and radical changes, this model might not be a fit.

2.2.9 State Based Workflow Models

The execution of activities inside the workflow usually imposes transitions from one state to another. If the workflow is strictly expressed as a set of subsequent predefined activities, then the execution of these activities should be completely independent of the state of the environment where they are executed. In real scenarios, this is not always possible. If the execution of activities depends on the environment state, then conventional workflow model is restrictive. Another approach to model a workflow would be, a set of activates that their execution depends on the definitions of state transitions using event and state based business rules [24]. In this case, the workflow might have many possible execution paths. This is a
better representation of a business process in real life scenarios when the environment is event driven and dynamic.

In the previous section was pointed out that conventional workflow systems are widely used as they are simple. On the other hand, state based workflow model provides more flexibility but obviously with a cost of complexity and usability.

Petri Net and Event Condition Action (ECA) analysed in the next sections are two main technologies that serve as platforms for state based business rule definition and state based workflow modelling. Both Petri Net and State Chart are two workflow models that facilitate process optimisation by introducing dynamic execution and flexibility. They can be easily adapted to new environment conditions on runtime. AOW framework adapts ECA model, being so a fit for an event drove environment such as telecommunication (discussion in section 1.1). Petri Net was considered to resolve an issue with parallel execution of activates, but the combination of these two models was impossible and out of the scope of the project. This discussion is detailed in section 3.8.3.

2.2.10 Petri Net

Petri Net model provides both the tool to model workflows and the mathematical theory to check the correctness of workflow execution. The designing capability is supported by three main concepts, Places, Arcs and Transitions (PAT). [25]

- **Places** that are represented by circles express a state or a condition.
- **Transactions** usually defined as rectangles are the activities of the workflow. Their execution will impact the state of the workflow.
- **Arcs** are used to link a place with a transition. They define the control flow.

A workflow modelled by Petri Net is called *Workflow Net*. The set of places and transitions linked together by arcs represent the workflow model as a whole. The Workflow Net can be described as possible routes that lead the input state to an output state. [26]

Possible routeing with Petri Net:

- **Sequential routeing**: Figure 2.4 defines a set of transitions that are executed one after the other. c1 represents the input state. The places c2 and c3 represent conditions that in this case are the execution of the previous transition. [27]
Parallel routing: Figure 2.5 defines a parallel execution of transition B and C. This happens when there is not any execution dependency between these two transitions. As the Figure 2.5 illustrates, AND-split and AND-join building blocks enable this type of routing. [27]

Conditional routing: Figure 2.6 allows the modelling of cases. The building blocks are OR-split and OR-join (exclusive OR). When the control flow is at c2, it will execute B or C, but they should never be executed together. In this case, the routing is done based on the condition c2 that might be a workflow attribute or environment state. [27]

Explicit choice: Figure 2.7 allows the modelling of cases defined by Workflow Variable x. Transition A leads in two places c2 and c3 based on x value. [27]

Petri Net Workflow System examples will be provided in section 2.3
2.2.11 Event Condition Action (ECA) state chart

State Chart (known as Event Condition Action) technology defines the state based business rules based on the principles of the state machine theory. This methodology defines the control flow and execution paths using Event – Condition – Action (ECA) rules [24]. Each state transition is defined based on the ECA rules specifications that are translated into the below if-then statement:

*If event (E) is executed and controls (C) are met then initiate action (A)*

Subsequently, the set of ECA rules defines both the control flow and data flow of the business process being modelled.

- Event (E) is represented by the execution of an activity. The result of this execution is communicated back to the workflow.
- Conditions (C) are set over data variables. These variables might be changed by the results of an event execution.
- Action (A) may generate another Event (E) or another Condition(C) [28].

On the top of this schema there are two additional important concepts, (1) the Nested States and (2) Orthogonal Components [24]:

- Nested State feature allows the organisation of workflows as state charts that might contain many sub-state charts enabling transitions from one state level to a subset of states in a lower level.
- Orthogonal Components feature allows the state charts to be executed in parallel in the same state level.

Examples of ECA workflow management systems will be provided in section 2.3.

2.2.12 The need for non workflow BPM methods

Even though the existing BPM tools are based on the traditional workflow concept, it does not mean that this is the best way to cope with the agility of the enterprise. In this context, agility refers to the ability of the business processes to be adapted to changes and constantly be improved. It is considered to become one of the most important characteristics of an enterprise in a dynamic and competitive market. As a result, these systems will not be able to cope with the frequently changing business needs. [29]
Researchers are looking for alternative ways of designing and modelling the business processes in a non-workflow based approach to enable the next generation of the enterprise to operate with the required agility. They are evaluating the possibility to introduce in BPM two completely independent concepts, *Systems Thinking* and *Mathematical System Theory*:

- **Systems Thinking**

  This theory in BPM [30] explains that the Enterprise System observes the changes happening from different internal and external interactions referred as events. This approach is illustrated in Figure 2.8. The changes might trigger Business Process Instance as Respondent Subsystem of the enterprise system. It will be terminated when the system reaches a predefined state or completes its goal. The Respondent Subsystem is controlled by the System Controls that defines its behaviour. This non workflow model is more dynamic than the workflow based ones. It responds to system changes and adjusts its behaviour accordingly on run time [31].

![System Thinking Theory in BPM](image)

Figure 2.8 | System Thinking Theory in BPM (Source: Ilia Bider, Stockholm University, 2012)

- **Mathematical System**

  This theory [32] in BPM considers the Business Process instance as an execution point that follows a trajectory in state, space, and time dimensions. Based on this point of view, the Business Process model can be expressed by a formula that provides all valid trajectories of this Business Process Instance. When the deviation between these trajectories is normal than this formula can be represented by standard workflow rules. Otherwise, in case that the
system state changes constantly and the possible trajectories differ significantly than only non-prescriptive rules can properly define them. [31]

Both these two theories aim in providing dynamic modelling and execution of the business processes. It can be observed that, even though these approaches are named non – workflow based models, they still follow and execution flow. However, this way of modelling removes dependencies such as static Input /Output or static order of actions (steps) execution. They seem to be an extensive and even more flexible version of existing state based workflow models. They are still under research, but most probably these state of the art models would be the foundation of the next generation BPM tools.

2.3 Existing Solutions

This section provides details of different existing solutions that are based on conventional workflow and state based models. These solutions are Workflow Management Systems that might model scientific or business processes. In addition, in these sections are analysed two BPM tools one from the heavyweight category and one from the lightweight category.

2.3.1 Existing Solutions based on Conventional Workflow model

As already discussed solutions based on Conventional Workflow model are very popular as they are simple to understand and use. Below are displayed two Workflow Management Systems that are used for scientific workflows. They model scientific experiments in different research areas [33].

- **Apache Taverna** is a Suite offering the tools to model and run scientific workflows. It operates in an SOA environment enabling the execution of different web services. It defines the workflows using SCUFL2 (Simple Conceptual Unified Flow Language version 2) which is an XML-based language [34]. It offers a GUI and a Command line to design and execute workflows. In addition, it supports parallelism and workflow scheduling. Tavern encourages reusability by workflow sharing.

- **Galaxy** is another popular example of conventional workflow management systems used to model scientific workflows. It provides as web-based GUI to model the workflows. It provides workflow sharing and scheduling capabilities as well [35].
2.3.2 Existing Solutions based on Event Condition Action model

Event Condition Action model is getting attention in Internet of Things and Ambient Intelligence scenarios. Many tools enable process automation by defining trigger – action rules. Below are provided two solutions that are adapting ECA [36]:

- **IF-This-Then-That (IFTTT)** is a tool used for automation. It provides a web and mobile application with a graphical interface. The user can define rules in the form of if then else statement which is a simplified version of ECA. Both the triggers and actions represent a predefined Web Service execution or a call to a smart device.

- **Atooma** is a mobile application very similar to IFTTT. It defines trigger – action rules, where an action can be a call to a smart device or service in one of 5 predefined categories. It is used for process automation in Internet of Things.

These two solutions are dynamic and user-friendly, but their applicability is limited in a specific area.

2.3.3 Existing Solutions based on Petri Net

As already discussed in section 2.2.10 Petri Net is used to model dynamic workflows. Below are displayed two existing BPM systems that are based on Petri Net workflow modelling.

- **Yet Another Workflow Language (YAWL)** is a BPM workflow management system that operates in SOA. Its main architectural components are YAWL Engine and Process Designer (Graphical interface). Modelling the workflows using Petri Nets allows this system to offer flexibility regarding [37]:
  - Representing real world activates by capturing deviations
  - Evolving the workflow modelling
  - Dealing with process flexibility
  - Offering extensive error handling capabilities by applying Petri Net theory.

- **COSA BPM** [38] is another example of a Petri Net Based BPM solution which operates in SOA. It offers workflow modelling, verification, monitoring, simulation, and optimisation as well as document management.

Both this systems offer extensive and sophisticated flexibility which is exposing a cost in complexity for both workflow modelling and workflow verification. These systems do not encapsulate properly the complexity associated with Petri Net theory.
2.3.4 Existing BPM tools

There are many BPM solutions currently in the market. Most of them are commercial and are offered as a full Suite, operating in SOA. In this section, we are going to analyse two BPM tools, one from the heavyweight category and the other from the lightweight.

**Heavyweight Category:** Oracle BPM Suite offers full capabilities of BPM in the enterprise. It is addressing all different user categories, and various aspects of BPM, from organisational and strategic to operational [7]. Figure 2.9 illustrates the application architecture of the BPM Suite and how it interacts with different actors and systems in an enterprise environment.

![Oracle BPM Suite Application Architecture](Source: Mark Wilkins, 2010, BPM Conceptual Architecture)

It provides two main modules, (1) design-time and (2) run-time which share the same repository. As all BPM tools do, Oracle offers a repository to store the BPM models and processes. Design-time module provides the environment to design the Business Process workflows and to store them in the repository. It also provides the different BPM models as reusable components. These components represent domain operations with high level of abstraction. The workflow modeller still needs to adjust these components to use them in a specific workflow. The Run-time module offers a variety of functionalities that covers every enterprise needs. Considering the architecture on Figure 2.9, we can conclude on the below advantages and disadvantages of this tool.
<table>
<thead>
<tr>
<th>Advantages</th>
<th>Disadvantages</th>
</tr>
</thead>
<tbody>
<tr>
<td>Stable and Robust Solution</td>
<td>Expensive solution</td>
</tr>
<tr>
<td>Highly adaptable</td>
<td>Complex solution. Need extensive training</td>
</tr>
<tr>
<td>High Availability and Security</td>
<td>Not flexible Change Control Procedures</td>
</tr>
<tr>
<td>Offers almost all possible functionalities</td>
<td>Consumes many resources on runtime</td>
</tr>
<tr>
<td>Supports all possible integration</td>
<td>Is supported by a complicated infrastructure</td>
</tr>
<tr>
<td>Serves to different user categories</td>
<td>Not a fit for simple enterprise infrastructures</td>
</tr>
<tr>
<td>Offers component Reusability</td>
<td></td>
</tr>
</tbody>
</table>

Table 2.3 | Advantages and disadvantages of Oracle BPM Suite

*Lightweight Category:* IF-This-Then-That (IFTTT) discussed in section 2.3.2 is not a dedicated BPM suite like Oracle solution, but it might be adapted and used for BPM if the architecture of the environment is not complicated. IFTTT automates different processes using triggers. It accesses the various services via Amazon Web Services (AWS). It has a simple and intuitive modeller and is simple to be used. It is a rule based tool that works over ECA. Table 2.4 lists the advantages and disadvantages of IFTTT as a BPM tool in Enterprise. [39]

<table>
<thead>
<tr>
<th>Advantages</th>
<th>Disadvantages</th>
</tr>
</thead>
<tbody>
<tr>
<td>Simple architecture</td>
<td>Does not offer high availability</td>
</tr>
<tr>
<td>Highly maintainable</td>
<td>Security is not a prime concern</td>
</tr>
<tr>
<td>High flexibility</td>
<td>Might have performance issues on high demand</td>
</tr>
<tr>
<td>Easy to get started and used</td>
<td>Offers limited design features for an enterprise</td>
</tr>
<tr>
<td>A good fit for simple process automation</td>
<td>Offers Limited integration capabilities</td>
</tr>
</tbody>
</table>

Table 2.4 | IFTTT advantages and disadvantages as BPM tool
2.3.5 AOW position among existing solutions

In the previous subsections were discussed different solutions that fall into different categories of workflow models and have different capabilities.

<table>
<thead>
<tr>
<th>Conventional</th>
<th>Apache Taverna</th>
<th>Galaxy</th>
<th>IFTTT</th>
<th>Atooma</th>
<th>YAWL</th>
<th>COSA BPM</th>
<th>AOW</th>
</tr>
</thead>
<tbody>
<tr>
<td>ECA</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Petri Net</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Table 2.5] Positioning of AOW among existing solutions. a) Categories of workflow models. b) Capabilities of existing solutions

Table 2.5 shows a synthesis of these systems capabilities. It also illustrates the characteristics of AOW, providing view where AOW sits among these solutions. Part a) of this table categorises the solutions based on the workflow model. As already stated AOW adapts ECA state based workflow model. Part b) of this table illustrates the capabilities that these systems have or miss. AOW capabilities are provided as well. Being a centralised, lightweight and reusable BPM solution is the aim of this project. These capabilities are analysed in Chapter 3 and are evaluated in Chapter 5. AOW is not a BPM Suite yet, as it is a pilot. Part of the future work will be to convert this pilot in a complete BPM suite.

2.4 Summary

In this chapter are defined system integration in the enterprise, BPM and Workflow models, and existing solutions.

I. Integration:

- An overview of SOA is provided by emphasising its *centralisation* and *reusability* features. AOW framework operates in an SOA environment (explained in section 3.6).
- ESB and Web Services are detailed as integration mechanisms of SOA. Talend and RESTful web services are defined as the best integration option regarding *lightweight*. Analysis of these technology options is synthesised on Table 2.1 and on Table 2.2.
Therefore, technology choices for the integration layer of AOW are performed based on these results.

II. BPM in Enterprise:

- A high level architecture model of a BPM solution in an SOA environment is defined. It serves as input model for the high level architecture of AOW framework detailed in section 3.5.
- It details analysis of BPM approaches. It is focused on the enterprise agility. Agility meaning is explained in section 2.2.12. The analysis of different BPM approaches are synthesised in Table 2.6.

<table>
<thead>
<tr>
<th>BPM approaches</th>
<th>Non Workflow</th>
<th>Conventional Workflow</th>
<th>State based Workflow</th>
</tr>
</thead>
<tbody>
<tr>
<td>Flexibility and Agility</td>
<td>High Agility</td>
<td>Not very flexible</td>
<td>Very Flexible</td>
</tr>
<tr>
<td>Complexity</td>
<td>High</td>
<td>Low</td>
<td>Moderate to High</td>
</tr>
<tr>
<td>Availability</td>
<td>state of the art: under research</td>
<td>off the shelf models</td>
<td>off the shelf models</td>
</tr>
</tbody>
</table>

Table 2.6 | BPM approaches characteristics

Referring in section 2.2.12, non workflow BPM models are described as the best approach that operates with the agility and flexibility required from the enterprise, but these models are under research. Therefore, this project is based on the best available option for business processes modelling regarding flexibility, which is the State based Workflows (ECA).

III. Comparison with existing solutions

This chapter analysis existing solutions in conventional workflow, ECA and, State based Petri Net models. Some of these solutions are used to model Scientific Workflows. Oracle BPM Suite is detailed as a heavyweight tool, while IFTTT as a lightweight one. A synthesis of these systems characteristics along with AOW capabilities is illustrated in Table 2.5.
Chapter 3. AOW Framework Design

3.1 Chapter Overview

This chapter discusses the high level architecture and design of the Action Oriented Workflow (AOW) framework. As already stated in section Chapter 1, AOW is a workflow based BPM framework for the enterprise that works as a state machine using Event Condition Action (ECA) state based model discussed in section 2.2.9. Early scenarios definition is provided to help the understanding of how AOW works. To deliver this project, an incremental and iterative approach has been followed.

The high level architecture of the system is based on the general BPM architectural model provided in section 2.2.6, including the two important phases of Design Time and Run Time. Graphical Designer Interface and Monitoring Tool are removed from the initial architecture to fit it in the project timeline and scope.

AOW operates in SOA. This architecture defines an integration layer implementing a RESTful web client and RESTful web server to consume and provide services respectively. In section 2.2.23 RESTful web services are defined as a more lightweight technology than SOAP ones. Talend ESB analysed in section 2.2.5 is introduced to translate other protocols to REST. Again, this technology choice is based on lightweight criteria.

Design Time defines the configurations on Data Model, which is made up of the Components Model and Rule Definition Model. As the Graphical Designer Interface is missing, all these configurations are defined directly in the Data Model Repository. The most important components of the Component Model are Flow, Action and Framework Command. Flow parameter values represent the flow state. Framework Command is a specific implementation of a Service. The Service itself is a generic implementation of operations that are executed locally or remotely. Both Pick Flow and Next Action algorithms are implementing ECA state based model. Petri Net model detailed in section 2.2.10 was analysed to resolve the concurrency management issue caused by action parallel execution, but it resulted that it was not addressing the issue and was introducing complexity. Therefore, Petri Net model is not used.

Run Time is composed of Workflow Initiation and Workflow Execution phases. In the first phase, once a trigger is received the Pick Flow algorithm selects the Flows that should be
instantiated and insert them in a *Flow Queue*. The second phase consumes the Flows from the queue and executes them, each Flow in a separate thread. *Thread pooling* is implemented to increase performance in multithreading and to reduce memory consumption.

Workflow Initiation and Workflow Execution phases have frequent communications with the Data Model in the database. Thus, *database connection pooling* is used to increase performance and response time.

### 3.2 AOW Framework Terms and Definitions

This project defines a variety of terms which have a specific meaning in this design. In Table 3.1 are provided the definitions of the most used terms in this chapter.

<table>
<thead>
<tr>
<th>Term</th>
<th>Definition</th>
</tr>
</thead>
<tbody>
<tr>
<td>Service</td>
<td>Is a generic implementation of operations that are executed locally or remotely</td>
</tr>
<tr>
<td>Flow Service</td>
<td>The generic service that picks the Flows and add them into Flow Queue</td>
</tr>
<tr>
<td>Mail Service</td>
<td>The generic service that sends Mails in TEXT and HTML format</td>
</tr>
<tr>
<td>REST Service</td>
<td>The generic service that consumes REST services using the RESTful Client</td>
</tr>
<tr>
<td>Database Service</td>
<td>The generic service that executes Database Store Procedures</td>
</tr>
<tr>
<td>Framework Command</td>
<td>A specific implementation of a generic Service</td>
</tr>
<tr>
<td>Triggering Framework Command</td>
<td>A Framework Command that implements Flow Service</td>
</tr>
<tr>
<td>Action Framework Command</td>
<td>A Framework Command attached to an Action</td>
</tr>
<tr>
<td>Hybrid Framework Command</td>
<td>An Action Framework Command that implements the Flow Service otherwise, a Triggering Framework Command attached to an action</td>
</tr>
<tr>
<td>Framework Command execution</td>
<td>It is the execution of the Service that the Framework Command implements</td>
</tr>
<tr>
<td>Environment Triggers</td>
<td>The triggers that might initiate Flows.</td>
</tr>
<tr>
<td>File Reader</td>
<td>Is an Environment Trigger. Is a Component that reads files and trigger workflows when specific conditions are met for a specific file record.</td>
</tr>
<tr>
<td>REST Events</td>
<td>Are Environment Trigger. They listen for a request that comes from external systems.</td>
</tr>
<tr>
<td>Pulse</td>
<td>Is an Environment Trigger. It is a timer that generates a trigger periodically. It is used to schedule workflows.</td>
</tr>
<tr>
<td>Action</td>
<td>It is an activity of the business process. It maps out a domain activity</td>
</tr>
<tr>
<td>Flow</td>
<td>It maps out the business process</td>
</tr>
<tr>
<td>Flow state</td>
<td>It is represented by the values of the Flow parameters</td>
</tr>
<tr>
<td>Pick Flow Algorithm</td>
<td>The algorithm that picks the Flows that should be initiated by a Triggering Framework Command based on ECA</td>
</tr>
<tr>
<td>Next Action Algorithm</td>
<td>The algorithm that picks the next Action that should be executed for the specific Flow based on ECA</td>
</tr>
<tr>
<td>Components Model</td>
<td>Includes Flow, Action, Framework Command and Parameter Mappings</td>
</tr>
<tr>
<td>Data Model</td>
<td>Includes Components Model, Pick Flow and Next Action Algorithm</td>
</tr>
<tr>
<td>Data Model Repository</td>
<td>Represents the database schema that defines the data model.</td>
</tr>
<tr>
<td>AOW File Repository</td>
<td>The files that File Reader processes are stored in this repository.</td>
</tr>
</tbody>
</table>

*Table 3.1| AOW Framework Terms and Definition*
3.3 Scenarios Definition

One of the challenges of this project is the generalisation. As we will see in the upcoming sectors, many design decisions are related to this feature which enables *reusability*. However, it has a related cost in complexity. In this section a definition of construction scenario and evaluation scenarios is provided. This early scenarios definition helps to keep this project focused. In addition, it helps in generalising the design.

There are two types of scenarios:

- Construction Scenario is used during the design and implementation of the system to test it.
- Evaluation Scenarios are used to evaluate the pilot against the project aim and objectives.

On section 1.1 is discussed that AOW fits well in Telecommunication as it is a typical dynamic and event driven environment. Therefore, all the scenarios will be defined in this environment. Terminologies of the systems in this environment have been defined in section 1.1.

*Construction Scenario*

*Customer Base Segmentation (CBS)* promotions are bonuses attached to customer segments (list of clients). Below is provided the Construction Scenario with Scenario Name: Lazarus Promotion

*If the customer performs a Balance Top Up with 5 Pounds or more and is part of “Inactive Customers” segment, then s/he will get a bundle promotion “ABundle” with 200 national minutes and 100 national SMS valid for one month. The customer profits this promotion only once per month. The customer should get a notification SMS after the successful bundle subscription.*

This scenario is translated into below table:

<table>
<thead>
<tr>
<th>No</th>
<th>Scenario Name</th>
<th>Trigger</th>
<th>Segment</th>
<th>Bonus</th>
<th>Bonus Type</th>
<th>Minimum Recharge Amount</th>
<th>Maximum Recharge Amount</th>
<th>Promotion Frequency</th>
<th>SMS</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Lazarus Promotion</td>
<td>Balance Top Up</td>
<td>Inactive Customers</td>
<td>Abundle</td>
<td>Bundle</td>
<td>5 Pounds</td>
<td>no max</td>
<td>1</td>
<td>yes</td>
</tr>
</tbody>
</table>

*Table 3.2| Lazarus Promotion Scenario translated into tabular form*

The trigger represents the event that the customer should perform to get the bonus. There are two bonus types Bundle (promotions grouped together) and Bucket (only one promotion).
The recharge amount should be in the interval defined by Minimum and Maximum Recharge Amounts to get the bonus. Promotion Frequency defines the number of times the customer can get the bonus in a month.

The triggering of the workflow is the event of Balance Top Up illustrated in Figure 3.1.

![Figure 3.1 Balance Top Up Event Flow](image)

The balance of the customer is increased on Billing, and this event is recorded in the Files produced by this system. These records are filtered and stored in AOW File Repository. Once the event is identified, the workflow that gives the bonus is initiated. It is illustrated below:

![Figure 3.2 Lazarus Scenario Workflow model](image)

This workflow has been simplified. Please refer to Appendix A for full workflow modelling and configurations. Firstly the workflow checks if the customer is part of “Inactive Customers” segment. The customer segmentation is performed in CRM. After that, it gets the subscriber data in Billing. Next Get Promotion Data checks the promotion rules like: is the recharge amount > 5 pounds; has the customer profited the bonus for that month; has the customer this bonus in his profile; which is the bonus type. This custom logic is performed in
Database. Based on the results of this operation the workflow might Remove Bundle/Bucket and then Subscribe Bundle/Bucket or directly Subscribe Bundle/Bucket. These last operations are performed in Billing. After Successful bonus subscription, the workflow sends to the customer a notification SMS. In this chapter will be defined how this workflow is translated into a Data Model and how is it going to be executed.

*The evaluation scenarios :* For the evaluation of the project will be used in total 6 scenarios which are categorised into two groups:

- In the first group are 5 CBS promotions.
- In the second Group is a Loyalty program scenario where the customers collect points from different events and redeem these points to get bonuses and gifts.

These scenarios will be detailed in section 5.2.

### 3.4 Methodology

The methodology that is used in this project is an incremental and iterative approach. Figure 3.3 lists the incremental steps and the iterations that are performed to deliver this project.

![Figure 3.3 | Methodology: An Incremental and Iterative approach](image)

The first architecture is designed to point out a complete Action Oriented Workflow framework serving as a BPM suite in an enterprise environment. Due to time limits, the aim of this project is focused only on the main modules of BPM engine. As a result, a simplified
version of this architecture is created to drive the low level design and implementation of the pilot. The implementation phase goes through iterations where evaluations are performed for the purpose of improvement.

3.5 Initial Framework Architecture

The general architectural model provided in Figure 2.3 on section 2.2.6 serves as an input for the high level architecture of the Action Oriented Workflow (AOW) framework provided below. Figure 3.4 illustrates the full architecture of AOW as a BPM Suite.

![Diagram of AOW as a BPM Suite](image)

**Figure 3.4 | High Level Architecture of Action Oriented Workflow Framework as a full BPM Suite**

We can identify all the modules of the general architectural model:

- **Business process modelling** is represented by the Design Time modules (Workflow Modelling Engine and Rule Definition Engine). These modules are supported by a Graphical interface to facilitate the configuration process. This interface provides capabilities like workflow configuration guidance, workflow deployment and versioning, workflow simulation and monitoring.

- **Business Process Environment** is represented by the Environment module. It is responsible for generating the triggers of the business processes (workflows). It communicates with Run Time modules.
- **Process Engine** is the Run Time Environment. It provides the workflow execution environment along with logging and monitoring capabilities.
- **Business Process Model Repository** is the Data Model Repository, which is made of Components Model, Pick Flow and Next Action algorithms. The Workflow Modelling Engine defines the Components Model. While Rule Definition Engine defines the rules for Pick Flow and Next Action algorithms.
- **Service Providers** module is represented by the combination of Communication module (RESTful Web Server and RESTful Web Client) discussed in section 2.2.12 and Talend ESB mentioned in section 2.2.4. Talend ESB jobs adapt incompatible communication interfaces. For example, if an Information System provides services via SOAP, then we need to create a Talend job that translates a RESTful request to a SOAP one.

### 3.6 Framework Architecture Review

Figure 3.5 illustrates the new architecture of the system which is less complex than the initial one.

The implementation of the architecture illustrated in Figure 3.4 can not be addressed in this project plan schedule. In addition, it demonstrates a full BPM suite, while the aim of this
project is focused on the BPM Engine. Finally, this architecture is too complicated for a pilot implementation. For these reasons, this architecture has been reviewed and simplified.

The Graphical Design Interface is removed while Design Time module is simplified. Components Configuration defines the Components Model (Flow, Action, and Framework Command). There are three types of Framework Commands:

- **Triggering Framework Commands**: In the Workflow Initiation Phase, the Triggers that come from different directions (RESTful Server, Pulse and File Reader) are translated to Framework Command using Triggering Framework Command Adapters. These Commands are referred as Triggering Framework Commands.
- **Action Framework Commands**: In Workflow Execution Phase an action might request the execution of a Framework Command. These Commands are referred as Action Framework Commands. Action Framework Command adapters are used to translate the response to the service execution to a Framework Command.
- **Hybrid Framework Command**: a Framework Command attached to an action that trigger flows.

Rules Configurations defines the rules that Pick Flow and Next Action Algorithm need to operate. In the absence of the graphical design interface, these configurations are directly inserted in the Data Model Repository. This Data Model is detailed in section 3.8.

At Run Time, the extended mechanism of workflow monitoring and reporting is not included in this version. The logging mechanism is used mainly for debugging. The Workflow Engine is composed of *Workflow Initiation* phase and *Workflow Execution* phase.

Workflow Initiation Phase selects the flows that should be instantiated by an environment Trigger and add them in the flow queue. Pick Flow Algorithm is executed in this phase. This phase is analysed in section 3.9.

Workflow Execution phase, consume the flows from the queue and executes them in separated threads. This phase is analysed in section 3.10.

The Communication Layer (RESTFful web services) and Talend ESB are used to integrate the framework with the other systems in the enterprise. As already discussed in Chapter 2. This design choice enables *Centralisation* and *Lightweight*. 
The Environment Component is responsible for receiving or producing the triggers and forward them to Workflow Engine Component. For the moment there are three types of triggers supported by the system:

- **REST Events**: the framework exposes a REST Web Service that can be accessed by different external systems. Based on the discussion on section 2.2.12 RESTful web services are considered to be more lightweight than other options like SOAP. In this context this design choice serves the aim and objectives of this project.
- **Pulse**: issues a periodical trigger that enables the execution of scheduled workflows.
- **File Reader**: is a process that reads files and processes every record. When specific conditions are met for a record, this component issues a Hybrid Framework Command that might initiate workflows. Table 3.3 illustrates the command that triggers the construction scenario workflow.

<table>
<thead>
<tr>
<th>Framework Command</th>
<th>Description</th>
<th>Parameters</th>
<th>Service</th>
<th>Is Executed in System</th>
</tr>
</thead>
<tbody>
<tr>
<td>File Reader Command</td>
<td>The command that is triggered from the file reader</td>
<td>Flow Names, Mileon, Recharge Amount, Event Label, Bundle name</td>
<td>Flow Service</td>
<td>AOW</td>
</tr>
</tbody>
</table>

Table 3.3| Construction Scenario, Hybrid Framework Command Configurations.

Even though there are only three integration options, it does not restrict the usability of the framework. In most of the cases, integrations introduce modifications in the systems involved. If the existing systems can not introduce a REST client to consume the REST Service exposed from AOW, then Talend ESB can be used to convert different requests to REST ones. Thus, in general, the integration of the framework in a specific enterprise environment should not be an issue regarding triggering.

### 3.7 AOW Framework Services

Service in AOW framework represents a generally configurable operation. In Design Time the Framework Command is configured to implementation a generic service. In Run Time the service is initiated from a Framework Command Execution, and it is executed locally or remotely.

Currently, AOW supports below services:

- **Flow service** is responsible to pick the flows (by executing Pick Flow Algorithm) and add them to the flow queue.
- *Database* is responsible for executing a database store procedure
- *Mail* is responsible for sending emails in plain text or HTML format
- *REST* is responsible for calling a rest service published by another system.

Service reusability is a system requirement, which is achieved by designing the services in a generic way. This has a related cost in complexity. Making the services, generic introduce many configurations that should be performed in Framework Command level. One of the challenges of this design was to strike a balance between generalisation and complexity in configuration level. As these are low level design details, this discussion will be described in detail in section 4.44 and 4.55 of Implementation Chapter.

Even though at the moment there are only four services, as we will see in the next chapter the service design is scalable to facilitate framework extension with new services.

RESTful web service, as a late technology is replacing SOAP and other communication protocols. Thus, having this service in place is a significant advantage. In addition, to overcome this system restriction, Talend ESB can be used to convert other protocols to REST.
3.8 Design Time

Design Time is the phase where a business process is translated into a workflow. This translation requires domain knowledge. Figure 3.6 illustrates most important components that need to be configured to translate a business process into an AOW workflow. It has two main parts, Components Configuration and Rule Configuration.

![Figure 3.6 AOW Data Model](image)

Components Configuration defines the Components Model which comprises of Flow, Action and Framework Command. While, Rule Configuration is represented by Rule Definition Model. It adapts ECA state based workflow model as they offer more flexibility comparing to conventional workflows (discussion in section 2.2.8). However, it exposes complexity in rules definition. One of the challenges at this point is to find a way to minimise and encapsulate this complexity.
3.8.1 Components Configuration, Components Model

Components Configuration phase defines the components illustrated in Figure 3.7.

![Components Model Diagram](image)

Figure 3.7 | AOW Data Model: Components Model

The components that are configured during components configuration are:

- **Flow** logically represents the business process. Technically it is directly attached only to the first action. An important concept is:
  - **Flow state which is represented by the values of Flow parameters.**

  Table 3.4 illustrates the Flow configurations for construction scenario.

<table>
<thead>
<tr>
<th>Flow</th>
<th>First Action</th>
<th>Status</th>
<th>Start Date</th>
<th>End Date</th>
<th>Flow Parameters</th>
</tr>
</thead>
<tbody>
<tr>
<td>CBS Bonus</td>
<td>Get Segment</td>
<td>Active</td>
<td>01.07.2016</td>
<td>01.07.2017</td>
<td>Msisdn</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Bonus</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Recharge Amount</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Bundles</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Buckets</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Segment Name</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Bonus Type</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Exists</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>SMS Text</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Command Result</td>
</tr>
</tbody>
</table>

Table 3.4| Construction Scenario, Flow Configurations

The flow has a Status (Active/Deactive), Start Date and End Date. These configurations are called flow profile. In sections 3.10.2 and Error! Reference source not found. is detailed how these configurations are used in the workflow execution logic.
The Triggering Framework Command parameters initialize the flow parameters based on the parameter mapping between Flow parameters and Framework Command parameters. In addition, flow parameters are updated by Action output parameters based on the parameter mapping between Flow parameters and Action parameters. Therefore, these initializations change the Flow state. For example, Msisdn and Recharge Amount are initialised from the File Reader Command parameters with the same name.

- **Action** maps out a domain activity. It is the one of the most important components regarding reusability. It has input/output parameters with default values. The input parameters are initialised by flow parameters. The output parameters are initialised from the action execution. An action should have *at least* one of the below components:
  - Input/Output BeanShell script
  - Input/Output Manipulator class
  - Framework Command

BeanShell scripting and Manipulator classes are Java dynamic code execution mechanisms that will be detailed in section 4.3.3. They are introduced in Action level to give the user the possibility to manipulate input and output parameters. Usually, the activity of the action is mainly performed by the execution of the Action Framework Command attached to it.

Each operation in the workflow in Figure 3.2 is mapped to an Action. For example, *Get Segment* is assigned into *Get Segment* Action defined in the table below.

<table>
<thead>
<tr>
<th>Actions</th>
<th>Description</th>
<th>Input Parameter</th>
<th>Output Parameter</th>
<th>Framework Command</th>
</tr>
</thead>
<tbody>
<tr>
<td>Get Segment</td>
<td>Checks in CRM if the Msisdn is in any Segment</td>
<td>Msisdn</td>
<td>Segment Name</td>
<td>Check Segment</td>
</tr>
</tbody>
</table>

Table 3.5| Construction Scenario, Get Segment Action Configurations

- **Framework Command** is a specific operation that can be performed using predefined Services. It contains the parameters and the operation of the Service it implements. Framework Command is a standard inside the internal logic of the framework. All the Environment Triggers are translated into Triggering Framework Command. This capability will be explained in more details in the next chapter.
Get Segment Action is executing Check Segment Action Framework Command. The table below defines the configurations of this command. It implements the REST Service

<table>
<thead>
<tr>
<th>Framework Command</th>
<th>Description</th>
<th>Parameters</th>
<th>Service</th>
<th>Is Executed in System</th>
</tr>
</thead>
<tbody>
<tr>
<td>Check Segment</td>
<td>Checks in CRM if the MSISDN is in any Segment</td>
<td>Msisdn</td>
<td>REST Service</td>
<td>CRM</td>
</tr>
</tbody>
</table>

Table 3.6 Construction Scenario, Action Framework Command Configurations

### 3.8.2 Rule Configuration, Pick Flow Algorithm

Pick Flow algorithm defines which group of flows will be initiated when a specific Triggering Command is executed. This algorithm adapts the ECA state chart model discussed in Background chapter 2.2.911. In this case:

- Event (E) is represented by the Environment Triggers that are listened. As already mentioned the triggers that the system supports in this version are Pulse, File Reader and Rest Events. These events are translated into Framework Commands (Triggering Framework Commands).
- Conditions (C) are the rules over Triggering Framework Command parameters. (Figure 3.6 Framework Command Rules)
- Action (A) will initiate the group of flows that meet the conditions.

Nested States concept is not considered while Orthogonal Components (parallel execution) is applied by allowing multiple flows to be executed in parallel, each flow in a separate thread. This algorithm is executed each time Triggering Framework Command is executed. Even though each flow is executed independently, there might be scenarios where a flow should not be performed if an instance of itself or other flows are being executed. Flow profile configurations have been introduced to manage these scenarios.

Pick Flow Algorithm operates over Triggering Framework Command Rules.

<table>
<thead>
<tr>
<th>Framework Command</th>
<th>Framework Command Parameter</th>
<th>Pattern</th>
<th>Flow</th>
</tr>
</thead>
<tbody>
<tr>
<td>FileReader Command</td>
<td>Flow Names</td>
<td>,CBS Bonus</td>
<td>CBS Bonus</td>
</tr>
</tbody>
</table>

Table 3.7 illustrates the Pick Flow rules configured for the construction scenario. *File Reader Command* initiates *CBS Bonus* Flow only when command parameter *Flow Names* contains “CBS Bonus” text.
3.8.3 Rule Configuration, Next Action Algorithm

Next Action is the algorithm that enables the workflow to operate as a state machine.

Again this algorithm is based on ECA state chart model where:

- Events (E) are generated when Flow state is changed. In other words, it happens when the flow parameters are modified after the execution of the current action.
- Controls(C) are the rules based on flow parameter values. (Figure 3.6 Next Action Rules)
- Action (A) represents the execution of the next action defined by the rules.

When the Flow parameters are updated by the Action, a Flow state change is triggered. The next action is defined based on Next Action Rules.

3.7.3.1 Next Action Algorithm Challenges

During the design of this algorithm, challenges were identified. Both options related to a design decision had different problems to be addressed. These options and related challenges are described below:

- Allow action parallel execution. That means the algorithm might return *more than one next action*. In this case, the challenge is how to manage concurrency when an action tries to update the flow parameters.
- Only one action can be executed at a time in a workflow instance. That means the algorithm *returns only one next action*. The user should avoid configuring ambiguous rules that might trigger more than one action. Defining next action rules might be already a complicated task depending on how complex the business rules are. Adding this constraint makes the rules definition configurations even more difficult and error prone. The challenge in this case is exactly how to make the rules definition more straightforward by reducing the possibilities of configuration errors.

Petri Net was evaluated to address problems with parallel execution, but it was not achieved. More particularly the parallel routeing and building blocks (AND-split, and AND-join) were assessed to be incorporated into the actual design. These concepts have been analysed in
section 2.2.910. After careful evaluation was decided that Petri Nets will not be used and action parallel execution will not be allowed for the below reasons:

- The combination of Petri Net and ECA state based models was too complex and out of the scope of this project.
- Parallel execution of Petri Net was not directly addressing the concurrency issue with flow parameter initialisation.
- Probabilities and priorities should be introduced to deal with the issue of flow parameter initialisation. These means, the user should perform more configurations which are complex and error prone.
- With the current design, each flow is executed in a separate thread. Allowing action parallel execution means that each action should be performed in a separate thread as well. It introduces a lot of changes on the current design and complexity in thread management and flows action communication.

So it was decided that Next Action Algorithm will return only one action. In case the conditions are met for more than one next action then the algorithm returns the first one. That means the user needs to take into consideration this system constraint when designing the Next Action Rules.

3.7.3.2 Next Action Rules Definition

To reduce the possibilities for rule configuration errors the algorithm is designed as per below figure.

<table>
<thead>
<tr>
<th>Current Flow</th>
<th>Current Action</th>
<th>Next Action</th>
<th>Rules Group</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td>Flow Parameter 1</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>Flow Parameter 2</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>Flow Parameter n</td>
</tr>
</tbody>
</table>

Figure 3.8 | Next Action Algorithm Design

- The rules are specific for the current Flow, the current Action and the next Action that the algorithm should return when the set of rules are satisfied.
- The rules are grouped together in Rules Group, meaning that all the rules in a group should be met in order the specific next Action to be initiated. Rules Group is unique.
- Actual rules are defined using regular expressions represented by Pattern.
In Figure 3.2, it is observed that to go from one action to the next one, conditions should be met. For example, to go from Get Segment to Get Subscribed Data, the segment should not be empty or otherwise zero. It is expressed with the below configurations.

<table>
<thead>
<tr>
<th>Flow</th>
<th>Current Action</th>
<th>Next Action</th>
<th>Flow Parameter</th>
<th>Pattern</th>
<th>Rule Group</th>
</tr>
</thead>
<tbody>
<tr>
<td>CBS Bonus</td>
<td>Get Segment</td>
<td>Get Subscriber</td>
<td>Segment Name</td>
<td>^[A-Z]</td>
<td>1</td>
</tr>
</tbody>
</table>

Table 3.8 | Construction Scenario, Next Action Rules Configuration

As we can observe, the rules are based on the value of Flow parameters. If the value matches the Pattern, then the rule is matched. Knowing that Flow state is represented by the parameter values, it is observed that the Next Action Rules are based on the Flow State.

Defining the rules in “Current Flow - Current Action” level narrows down significantly the rule space that the user needs to check. Rule Space for “Current Flow - Current Action” actually is the number of branches derived from the current action. In simple workflows it might be one or two. The maximum number of branches in the construction scenario workflow is four.

3.7.3.3 Next Action Algorithm restrictions

In the current version of the framework, the user can simulate the next action rules algorithm after inserting the new set of rules and check the results. This algorithm is general, offering so the possibility to map many business rules in Next Action Rules, but this generalisation always comes with a cost in complexity and usability.

Besides the fact that rule definition is a complicated task, this algorithm has some additional restrictions:

- Not all the business rules can be expressed in the regular expression. It highly depends on the complexity of the domain.
- The controls should be performed or simulated by the user, which is not a recommended behaviour. The system should automatically complete these checks.
- The user should have extensive knowledge of regular expressions to be able to define complicated rules.

Therefore this algorithm introduces issues in striking a balance between generalisation, complexity and usability.
3.9 Run Time: Workflow Initiation phase

Figure 3.9 illustrates the process of workflow initiation when a trigger is received.

The environment triggers are already described in section 3.6. As Figure 3.9 suggests, all these triggers are translated to Triggering Framework Commands using adapters. This translation will be detailed in 4.5.25. The triggering commands implement the Flow Service. Therefore, the execution of the Triggering Framework Command calls the execution of the Flow Service. This last one picks the Flows that should be initiated using Pick Flow Algorithm and puts these Flows into the Flow Queue. The response of this execution is again a Framework Command.

Flow execution might take a long time depending on the business process it automates. Therefore, each flow is executed in a separate thread. Thread pooling capability that will be detailed in next section is introduced to reduce the cost of thread allocation and to improve
performance. Thus, flow queuing is used to queue flow execution requests, and to submit these tasks to the thread pool.

On the other hand triggering requests are neither threaded nor queued as:

- The response to a trigger is returned once the flows are picked and added to the queue. This process normally takes a fraction of seconds to be executed.
- Usually, the triggering requests are not interested in getting feedback on the workflows execution. They are responsible only for the triggering of these workflows. Thus, the only information on the response is the workflows that are triggered and an error code.

### 3.10 Run Time: Workflow Execution phase

In the previous section is discussed how the flows are instantiated and added to the queue. In this section will be detailed the execution phase that consumes the flows from the queue and executes them. Figure 3.10 suggests four important modules that will be discussed in this section, (1) Flow Queue, (2) Flow Executor Manager, (3) Flow Executor, and (4) Action Executor. It expresses how these modules are linked together.

![Figure 3.10 | Workflow Engine control flow](image-url)
3.10.1 Flow Queuing

There are many queuing options available from advanced messaging middleware to simple queue collections. Table 3.9 synthesises some capabilities of three different queue options.

<table>
<thead>
<tr>
<th>Features</th>
<th>RabbitMQ</th>
<th>ActiveMQ</th>
<th>BlockingQueue</th>
</tr>
</thead>
<tbody>
<tr>
<td>Technology</td>
<td>Messaging Middleware</td>
<td>Messaging Middleware</td>
<td>Advances Collection Type</td>
</tr>
<tr>
<td>Concurrent Applications</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>Thread Safe</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
</tr>
</tbody>
</table>

Table 3.9 | Queues capabilities and features

RabbitMQ and ActiveMQ offers a set of sophisticated features like clustering or remote queue consumptions. In general, these are useful capabilities that improve the performance of the system, but they make the architecture more complicated. They are not just queues, but they are separate nodes working as messaging middleware.

*Thus, these two options are not considered as lightweight in the context of this project.*

On the other hand, BlockingQueue is an advanced queue collection in Java that has simple but useful features like:

- It blocks the consumer when the queue is empty, and blocks the producer when there is not available space.
- It works well in concurrent applications
- It provides a thread–safe implementation
- It works well with thread pooling.

In addition, it offers the option of Linked BlockingQueue, removing so the restriction of static queue size. Therefore, this technology is used in this project to queue the workflows as per Figure 3.11.

*Figure 3.11 | BlockingQueue producer and consumer*
There is only one producer “Thread 1” which is exactly the main thread of the application. On the other hand, the consumer is a separate thread that submits the flow to the Flow Executor Manager. Each flow is consumed in a separate thread. It is the point where blocking queue is combined with thread pool capabilities that will be discussed in the next subsection section.

3.10.2 Flow Executor Manager and multithreading

The consumer picks a flow from the queue and submits it to the Flow Executor Manager. The submission process is illustrated in Figure 3.12. The manager maintains a list of flows that are in execution. Adding the flow to this list represents the submission of the flow into the manager. When a new flow comes for submission, its profile is checked against this list. This check will be detailed further in section 4.3.1. If the flow is not allowed, it is added back into the Flow Queue. Otherwise, after submission, a Flow Executor is instantiated, and a thread from the pool is assigned to it.

![Diagram](image.png)

Figure 3.12 | Flow Executor Manager, Consume Flow
Multithreading and concurrence can be managed in two ways:

- One request – One thread: In this case, for each flow execution a thread will be allocated and deallocated.
- One request – Thread Pool: this capability uses the worker threads to serve multiple flow executions. For each request thread is created up to “core pool size”. After that, a thread is created if the queue is full. There is also a “maximum pool size” defining the maximum number of threads that can be created by the application.

AOW implements Thread Pooling mechanism. Obviously, it has many advantages over One request – One thread. In Table 3.10 are provided with the advantages of using Thread pool in comparison with One Request – One Thread.

<table>
<thead>
<tr>
<th>One Request – One Thread</th>
<th>Thread Pool</th>
</tr>
</thead>
<tbody>
<tr>
<td>Thread Creation and Destruction for each request, is time-consuming</td>
<td>Create threads once and reuse result in better performance</td>
</tr>
<tr>
<td>Thread Creation and Destruction for each request, consumes much memory</td>
<td>Create threads once and reuse result in better resource management</td>
</tr>
<tr>
<td>Denial of Service Attacks might cause memory overload</td>
<td>Protected by Denial of Service Attacks as threads are created up to a maximum pool size</td>
</tr>
</tbody>
</table>

Table 3.10 | Advantages of Thread Pool over One Request - One Thread

3.10.3 Flow Execution

Flow Executor Manager discussed in the previous sub section initialises a Flow Executor for each flow that is being submitted in the manager. Flow Executor process is illustrated in Figure 3.13.

![Flow Executor, workflow execution process](image-url)
Firstly the flow parameters are initialised from triggering Framework Command parameters. This initialization is performed based on the Flow Parameter – Command Parameter discussed in section 3.8.1.

Next, the Flow Executor calls the Action Executor to execute the selected action. Action Executor will be detailed in the next subsection. After that, Next Action Algorithm will decide the next action based on the Flow State. The flow execution terminates when there is not any next action. Logging is performed in each of the steps defined in the diagram.

The first loop of construction scenario workflow execution: “CBS Flow” parameters (Msisdn and Recharge Amount) are initialised from “File Reader Command” parameters with the same name. Next, the first action “Get Segment” is executed. After this execution, Next Action Algorithm checks if “Segment Name” flow parameter is initialized with a value. If true, “Get Subscriber Data” action is executed otherwise the flow is terminated.

3.10.4 Action Execution

Action Executor performs the execution of an operation. It performs the following steps:

- Initialize input parameters from flow parameters based on the mapping defined in the Configurations Model.
- Execute at least one of the three operations (Bean Shell, Manipulator Class, and Framework Command). Usually, an action executes a Framework Command.
- Initializes output parameters with the result of execution
- Update Flow parameters with output parameters based on their mapping.

The Action Framework Command executes one of the three services, REST, Database and Mail.

Execution of “Get Segment” action: First “Msisdn” action parameter is initialized from flow parameter with the same name. This action has only a Framework Command that executes a REST request in CRM using REST service. The response of this execution initialises “Segment Name” action output parameter. Finally, the action initializes “Segment Name” flow parameter with the value of its output parameter with the same name.
3.10.5 DB Connection Pooling

AOW framework frequently communicates with the Component Repository which is represented by Oracle Database. Error! Reference source not found. displays all the most frequent database operations and when they are performed.

<table>
<thead>
<tr>
<th>Database Operation</th>
<th>Frequency</th>
</tr>
</thead>
<tbody>
<tr>
<td>get Triggering Framework Command</td>
<td>for each trigger</td>
</tr>
<tr>
<td>pick Flows</td>
<td>for each trigger</td>
</tr>
<tr>
<td>get Action Framework Command</td>
<td>usually for each action execution</td>
</tr>
<tr>
<td>get response Framework Command</td>
<td>usually for each action execution</td>
</tr>
<tr>
<td>get Flow</td>
<td>for each flow execution</td>
</tr>
<tr>
<td>get next action</td>
<td>for each action of the flow</td>
</tr>
<tr>
<td>get File filters</td>
<td>For each file processing</td>
</tr>
</tbody>
</table>

Table 3.11 | Most Common database operations and related frequencies

Taking into consideration the high volume of triggers, the parallel execution of flows, and the large amounts of actions that are going to be executed at the same time, AOW needs to open many database connections.

Creating a DB connection is a costly task in terms of performance and resources. As a consequence, the traditional database connection management (one database operation – one connection) is not working for this case. Therefore, Database Connection Pooling technology is used to improve the performance of the system. It applies the same pooling philosophy as Thread Pool, where the connections are pre-created and reused. This pooling mechanism is defined and configured in Web Server level (Tomcat).
### 3.11 Summary

Below table summarises the most important design decisions and characteristics discussed in this chapter.

<table>
<thead>
<tr>
<th>Design Time</th>
<th>Run Time</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Components Configuration: Components Model</strong></td>
<td><strong>Workflow Initiation Phase</strong></td>
</tr>
<tr>
<td><strong>Services</strong></td>
<td>Environment triggers (REST event, Pulse, File Reader) are translated in Triggering Framework Command</td>
</tr>
<tr>
<td>Flow Service</td>
<td>Triggering Framework Command executes the Flow Service, which picks the Flows and put them into the Flow Queue</td>
</tr>
<tr>
<td>REST Service</td>
<td>Flow Queuing uses BlockingQueue as a lightweight technology choice</td>
</tr>
<tr>
<td>Database Service</td>
<td>Workflow Executer Manager manages multithreading with Thread Pooling</td>
</tr>
<tr>
<td>Mail Service</td>
<td>Flow Executor and Action Executor perform the execution of the workflow</td>
</tr>
<tr>
<td>Generic implementation</td>
<td>Database Connection pooling is used to improve performance and resource management</td>
</tr>
<tr>
<td>Extensible design</td>
<td><strong>Rules Configuration: Rule Definition Model</strong></td>
</tr>
<tr>
<td>Flow has its parameters</td>
<td><strong>Next Action Algorithm</strong></td>
</tr>
<tr>
<td>The value of Flow Parameters represents its flow state</td>
<td>Rules are defined over Flow State Parameters using regular expressions</td>
</tr>
<tr>
<td>Has input parameters</td>
<td>It returns only one action at a time</td>
</tr>
<tr>
<td>Has output parameters</td>
<td><strong>Framework Command</strong></td>
</tr>
<tr>
<td>Has Bean Shell</td>
<td><strong>Pick Flow Algorithm</strong></td>
</tr>
<tr>
<td>Has Manipulator Class</td>
<td>It is based on ECA</td>
</tr>
<tr>
<td>Has Framework Command</td>
<td>Rules are defined over Triggering Framework Command Parameters using regular expressions</td>
</tr>
<tr>
<td>It is a specific implementation of a Service</td>
<td><strong>Flow</strong></td>
</tr>
<tr>
<td>Triggering Framework Command</td>
<td><strong>Action</strong></td>
</tr>
<tr>
<td>Action Framework Command</td>
<td>Has input parameters</td>
</tr>
<tr>
<td>Hybrid Framework Command</td>
<td>Has output parameters</td>
</tr>
<tr>
<td></td>
<td>Has Bean Shell</td>
</tr>
<tr>
<td></td>
<td>Has Manipulator Class</td>
</tr>
<tr>
<td></td>
<td>Has Framework Command</td>
</tr>
<tr>
<td><strong>Workflow Execution Phase</strong></td>
<td><strong>Pick Flow Algorithm</strong></td>
</tr>
<tr>
<td>Environment triggers (REST event, Pulse, File Reader) are translated in Triggering Framework Command</td>
<td>It is based on ECA</td>
</tr>
<tr>
<td>Triggering Framework Command executes the Flow Service, which picks the Flows and put them into the Flow Queue</td>
<td>Rules are defined over Triggering Framework Command Parameters using regular expressions</td>
</tr>
<tr>
<td>Flow Queuing uses BlockingQueue as a lightweight technology choice</td>
<td>It returns only one action at a time</td>
</tr>
<tr>
<td>Workflow Executer Manager manages multithreading with Thread Pooling</td>
<td><strong>Next Action Algorithm</strong></td>
</tr>
<tr>
<td>Flow Executor and Action Executor perform the execution of the workflow</td>
<td>Rules are defined over Flow State Parameters using regular expressions</td>
</tr>
<tr>
<td>Database Connection pooling is used to improve performance and resource management</td>
<td><strong>Flow</strong></td>
</tr>
</tbody>
</table>

*Table 3.12| Summary of Chapter 3*
Chapter 4. AOW Framework Implementation

4.1 Chapter Overview

In this chapter are explained important implementation details of AOW modules. Firstly, it is provided a synthesised view of the most important technologies that are used to implement the framework and its integration layer. In addition, some Run Time modules analysed in Chapter 3, are detailed further to provide a better understanding of how they work.

One of the challenges of this project is the generalisation of the Services and the corresponding complexities introduced in Framework Command configurations. In this chapter is defined how this generalisation is implemented and which are the needed configurations. In addition, it is analysed how different messages (triggers and service responses) are translated to Framework Commands.

Other important modules of the system are detailed such as RESTful Web Server, File Reader, and Logging. Furthermore, system and workflow deployment procedures are described.

4.2 Technology choices

Table 4.1 illustrates that the technology choices are based on the arguments and comparisons outlined in Chapter 2.

<table>
<thead>
<tr>
<th>Technology</th>
<th>AOW module</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>RESTful Web Services</td>
<td>Integration Layer</td>
<td>They provide the communication layer between BPM Engine and ESB, or directly with other information systems. It is depicted as a new, advanced, and lightweight technology.</td>
</tr>
<tr>
<td>JAX-RS Jersey</td>
<td>RESTful web Server</td>
<td>It is a Java API for RESTful Web Services [40].</td>
</tr>
<tr>
<td>Java Architecture for XML Binding</td>
<td>RESTful web Server</td>
<td>It is used to bind Java objects to XML or JSON [41].</td>
</tr>
<tr>
<td>Jersey Client</td>
<td>RESTful web Client</td>
<td>It is used to create a client that consume Restful Web Services [42].</td>
</tr>
<tr>
<td>DOM Parser and XPath</td>
<td>RESTful web Client</td>
<td>They are Java Libraries, used to parse the XML response of a web Service and extract the needed information using XPath [43].</td>
</tr>
<tr>
<td>Talend ESB</td>
<td>Integration Support</td>
<td>It is used as the integration Layer of this framework. The advantages of this platform over the other options are displayed in Table 2.2. It is chosen mainly as lightweight technology.</td>
</tr>
<tr>
<td>Oracle 11g DBMS</td>
<td>Repository</td>
<td>In the context of this project, there is not any constraint on the choice of the Data Base Management System (DBMS). For the pilot version, Oracle 11g DBMS is used. This choice is based on the author experience on this tool.</td>
</tr>
<tr>
<td>Dynamic Web Application in Java EE</td>
<td>Core Application</td>
<td>Java is an open source, platform-independent programming language [44].</td>
</tr>
<tr>
<td>Apache Tomcat</td>
<td>Application Server</td>
<td>It is an open source application and web server which supports Java web technologies. It is well known for its lightweight capability [45].</td>
</tr>
<tr>
<td>ECA</td>
<td>BPM Engine</td>
<td>It offers flexibility and dynamic execution of workflows. Therefore it is chosen as the best approach among available options.</td>
</tr>
<tr>
<td>Linked Blocking Queue</td>
<td>Flow Queuing</td>
<td>It is straightforward and lightweight queue option compared to other Messaging technologies. Analysis performed in section 3.10.1</td>
</tr>
<tr>
<td>Bean Shell Scripting</td>
<td>Action parameter manipulations</td>
<td>It is a Java mechanism that allows the execution of scripts saved on files or database at a specific moment in runtime. It is a lightweight scripting option offered by Java platform [46]. This mechanism is used to manipulate input/output action parameters.</td>
</tr>
<tr>
<td>Thread Pooling</td>
<td>Multithreading</td>
<td>It manages the multithreading by reusing pre-created threads. It improves resource management and performance. Analysed in section 3.10.2</td>
</tr>
<tr>
<td>Database Connection pooling</td>
<td>Database Connections</td>
<td>It manages the database connections by reusing pre-created connections. It improves resource management and performance. Further analysis can be found in section 3.10.5.</td>
</tr>
</tbody>
</table>

**Table 4.1** The technologies used to implement AOW Framework

### 4.3 Run Time implementation details

In this section will be discussed implementation details of some Run Time modules.

#### 4.3.1 Flow Executor Manager

Flow Executor Manager consumes the flows from the queue. Figure 3.12 illustrates that the flow profile is checked before submitting it to the manager. As already stated in section 3.10.2, the manager submits the flows in a list. This list stores the Fows in execution. In Figure 3.12, “Check Flow Profile” step checks whether the flow is allowed to be submitted to the manager.

*The flow is submitted if below conditions are met:*

- Flow Status is Active
- The current date is between the start and end date.
The flow is not blocked by any other flow that is submitted to the manager.

*The flow is added back in the Flow Queue when:*

- Flow status is Deactive
- The current date is between the start and end date.

This option is used when a flow is temporarily deactivated for maintenance reasons. It will be explained further in section 4.9. This implementation has an issue as it does not limit the number of retries for a specific flow and might result in an endless loop or increase the flow queue significantly.

*In all the other cases the flow is discarded.*

### 4.3.2 Database Connection Pooling and Database Connection Manager

The Database Connection pooling capability explained in section 3.10.5 is managed by a Database Connection Manager. AOW offers the ability to be connected to more than one database. This manager maintains the Data Sources, which represents the connection pools. A Data Source configuration needs to be performed for each database that AOW is connected. Currently, it is connected only to Oracle Database. Please refer to Appendix B.1 for configuration details. The manager maintains a Hash Map where the key is the Data Source Name and the value is the Data Source object. When a connection is requested, the Data Source Name should be provided as input. In Appendix B.1 is demonstrated the code that creates a Data Source and adds it into the Hash Map.

### 4.3.3 Action Execution

Action Executer is the process that manages all the steps for action execution illustrated in Figure 4.1:

1. Initializes input parameters
2. Manipulates input parameters using Bean Shell Script
3. Manipulates input parameters using Manipulator Classes
4. Executes a Framework Command (FW Cmd)
5. Initializes output parameters with the result of execution
6. Manipulates output parameters using Manipulator Classes
7. Manipulates output parameters using Bean Shell Script
8. Update Flow parameters with output parameters
Steps 1 and 2 are performed based on Flow Parameter – Action Parameter mapping, defined in flow configuration level.

Bean Shell scripting and Manipulator classes are both dynamic code execution mechanisms. They are used to manipulate the input and output parameters of the action in case there is any incompatibility between:

- Action and FW Command interfaces
- Flow and Action interfaces

Bean Shell scripting is defined as part of the flow configurations and is specific for a Flow and an action. It is defined by the user when designing the flow and does not require a system restart.

Manipulator class is a Java class that can be called at a certain execution point by providing the class Name. It should be introduced when the system is adapted to a specific environment and is delivered as part of the source code.
4.4 Services and Service Objects

In this section are analysed the implementation details of the Services. This concept is important in the context of this project as it will be used to demonstrate the first level of reusability (Service Reusability). Therefore, the design and implementation of this capability are performed based on reusability requirement. In this section are detailed the current services supported by AOW framework, Flow, Database, Mail and Rest services.

4.4.1 Overview

The service concept has been already defined in section 3.7. In AOW a service is represented by a Service Object having two operations:

- Service authentication: the systems that communicate with the framework should have or provide an account with a username, password and a specific IP address. The password is encrypted with MD5, and the IP addresses are maintained in a White IP list. Therefore, system communication is protected by a security layer.
- Service execution will be detailed for each specific service object.

Service Configuration:

The services are configured in the data model of the framework as per below:

<table>
<thead>
<tr>
<th>Service Name</th>
<th>Service Class Path</th>
<th>Service Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>FLOW</td>
<td>fw.serviceobjects.FlowServiceObject</td>
<td>The service that reads the commands and issues the flows</td>
</tr>
<tr>
<td>DATABASE</td>
<td>fw.serviceobjects.DBServiceObject</td>
<td>The service that executes the DB Queries</td>
</tr>
<tr>
<td>REST</td>
<td>fw.serviceobjects.RestServiceObject</td>
<td>The service that executes Rest APIs to other systems.</td>
</tr>
<tr>
<td>MAIL</td>
<td>fw.serviceobjects.MailServiceObject</td>
<td>The service that sends mails</td>
</tr>
</tbody>
</table>

The service class path is defined in the data model. The system allows the dynamic execution of the service objects based on class name. This feature enables the Framework Commands to execute the services based on configuration. In this way, the services become reusable components of the framework. This mechanism will be explained in more details in section 4.5. Appendix B.2 illustrates the code of service execution.
4.4.2 Flow Service Object, workflow initiation phase

The Flow Service Object is responsible for picking up the flows based on the Framework Command and inserting them in the Flow Queue.

Flow Service execution process is part of the workflow initiation phase discussed in section 3.99. Figure 4.2 shows this process and its relation to the architecture of this phase. When a triggering Framework Command is executed, the workflow service object is called using the class path. The Flow service executes the Pick Flow Algorithm using as input the Framework Command and its parameter values. If this algorithm returns zero, then the process returns a response Framework Command with Result Message “NO_FLOWS_ARE.Selected”.

Otherwise, it retrieves Flow Objects from the component repository and adds them in the Flow Queue. Next, it returns a response Framework Command with Result Message “FLOWS_ARE.Selected” and the flow names.

Reusability in this service level is evident as all triggering Framework Commands reuses Flow Service.
4.4.3 Database Service Object

This Service is used during workflow execution phase by an Action Framework Command. The Framework Command needs to provide the below input:

- **Resource Name** represents the Data Source Name.
- **Query String** executes a store procedure in the Database with the below format:
  
  
  ```
  begin?:={package name}.[store procedure name](?,?);
  ```

- **Input Parameters list** stores the input parameters that the database store procedure needs

This service might execute the queries in different databases using the Data Source Name as explained in section 4.3.2.

The execution of this service returns a Database Cursor that is a table of values. Even though this design introduces constraints on how the user is going to define the store procedures, it is general enough to expose this service as a reusable component.

4.4.4 REST Service Object

REST Service Object provides a general REST client to consume different REST services exposed by other systems. The generalisation of this Service is a challenge as RESTful Web Services might have many possible:

- structures (media types)
- operations (methods)
- input types

For this reason, the below configurations are introduced to support different RESTful services.

<table>
<thead>
<tr>
<th>Configuration Type</th>
<th>Possible Values</th>
</tr>
</thead>
<tbody>
<tr>
<td>URI</td>
<td>specific service URI</td>
</tr>
<tr>
<td>Method</td>
<td>GET, PUT, POST, DELETE</td>
</tr>
<tr>
<td>Input</td>
<td>Path, Query, Matrix, JSON, XML</td>
</tr>
<tr>
<td>Consumed Media Type</td>
<td>application/plain_text, application/xml, application/json</td>
</tr>
<tr>
<td>Produced Media Type</td>
<td>application/plain_text, application/xml</td>
</tr>
</tbody>
</table>

This service supports all REST operations, input types and most used structures. Therefore it is general and can be reused in many Action Framework Commands.
On the other hand, this generic approach introduces configuration complexity. The user needs to define many configurations for each Framework Command that executes this service.

4.5 Framework Command and Message Transformation

Framework Command is a specific implementation of a service. It works as a standard format representing the services inside the framework. On the previous section generic service objects were analysed. While in this section is displayed how these generic services are used for specific service implementations by pointing out the reusability feature on service level.

4.5.1 Overview Framework Command:

The Framework Command is used both in Workflow Initiation Phase and Workflow Execution Phase.

Figure 4.3 illustrates the Framework Command Executer component that is used to execute a Framework Command in both cases. The command execution calls dynamically the related service object using the Service Name and System Name to get the Service Class Path.
The executeCommand() method implementation illustrated in Figure 4.4 suggests that the response of a service execution is again a Framework Command.

The Framework Command has a specific structure provided below:

**Request:**

*System Name | Service Name | Command Name | Command ID | Attributes [Name: Value].*

**Response:**

*Request System Name | Request Service Name | Request Command Name | Command ID | Attributes [CommandResult: {Success, Fail}, ResultMessage: {general execution feedback}, Other Attributes].*

In section 3.6 are presented three types of framework commands. Their characteristics are shortly summarised in Table 4.4 and are detailed in the following subsections.

<table>
<thead>
<tr>
<th>Command Type</th>
<th>Phase</th>
<th>Execution</th>
<th>Response</th>
<th>System</th>
<th>Services</th>
</tr>
</thead>
<tbody>
<tr>
<td>Triggering Framework Command</td>
<td>Workflow Initiation</td>
<td>Framework Command Executor</td>
<td>Framework Command</td>
<td>AOW</td>
<td>Flow Service Object</td>
</tr>
<tr>
<td>Action Framework Command</td>
<td>Workflow Execution</td>
<td>Framework Command Executor</td>
<td>Framework Command</td>
<td>AOW</td>
<td>Database Service Object, Mail Service Object, REST Service Object</td>
</tr>
<tr>
<td>Hybrid Framework Command</td>
<td>Workflow Initiation and Workflow Execution</td>
<td>Framework Command Executor</td>
<td>Framework Command</td>
<td>AOW</td>
<td>Flow Service Object</td>
</tr>
</tbody>
</table>

**Table 4.4| Framework Command Types**

### 4.5.2 Triggering Framework Command

Triggering Framework Command has below Structure:

**Request:**

*AOW | FLOW | Command Name | Command ID | Attributes [Name: Value]*
Response:

AOW | FLOW | Request Command Name | Command ID | Attributes |
CommandResult: {Success, Fail},
ResultMessage :{ FLOWS_ARE_SELECTED, NO_FLOWS_ARE_SELECTED},
SELECTED_FLOWS :{ Name of Flows Selected},
EXECUTE_TIME

All Triggering Framework Commands call the same Service Object which is Flow Service Object (FLOW). Thus, the only purpose of a Triggering Framework Command is to trigger flows.

Command Name and Attributes are provided by the Triggering Events. The translation from a triggering event to a Framework Command is performed by Triggering Framework Command Adapters. Table 4.5 illustrates which information is extracted to carry out the translation for each adapter.

<table>
<thead>
<tr>
<th>Triggering Framework Command Adapter</th>
<th>Command Name</th>
<th>Attributes</th>
</tr>
</thead>
<tbody>
<tr>
<td>RESTfulToFWCommand</td>
<td>First Path Parameter of URI</td>
<td>Query parameters defined in the URI</td>
</tr>
<tr>
<td>PulseToFWCommand</td>
<td>Pulse</td>
<td>TIME</td>
</tr>
<tr>
<td></td>
<td></td>
<td>TIME_MILIS</td>
</tr>
</tbody>
</table>

Table 4.5 | Triggering Framework Command Adapters

File Reader also triggers a Framework Command, but it does not use any adapter. File Reader process is implemented using the workflow mechanism. It is a system defined workflow that will be detailed in section 4.77.

Refer to Appendix B.3 for examples of the Triggers translated to a Framework Commands.

The reusability of Flow Service Object is evident, as all triggering commands will implement this service. In addition, a Framework Command that implements this service is a distinct component that can be reused in terms that the same framework command can trigger many flows by defining different rules. For example, Pulse Triggering Framework Command can initiate different flows that need to be scheduled at various times or intervals.

4.5.3 Action Framework Command

The Action Framework Command is attached to an action and is executed inside the action execution. The command parameters values are initialised by the action parameters.

To avoid extra configurations and complexity, it is not defined any mapping between action and command parameters. Thus, this initialization is performed based on the name of the
parameters. On the other hand, the user should take in consideration this constrain when defining actions in the data model.

An action Framework Command executes a service remotely to perform a specific operation. Therefore, this command implements one of the below services:

- DATABASE (Database Service Object)
- REST (REST Service Object)
- MAIL (Mail Service Object)

In section 4.4 it was already explained what these services are and how they execute their operations remotely. All these services have attributes that make them generic. When a command is implementing a specific service, it has to configure these attributes with specific values.

<table>
<thead>
<tr>
<th>Service</th>
<th>Attributes</th>
</tr>
</thead>
<tbody>
<tr>
<td>Database Service Object</td>
<td>Query String</td>
</tr>
<tr>
<td></td>
<td>Resource Name</td>
</tr>
<tr>
<td></td>
<td>[inputs of database store Procedure]</td>
</tr>
<tr>
<td>Mail Service Object</td>
<td>Sender Address</td>
</tr>
<tr>
<td></td>
<td>Receiver Address</td>
</tr>
<tr>
<td></td>
<td>Subject</td>
</tr>
<tr>
<td></td>
<td>Content</td>
</tr>
<tr>
<td></td>
<td>[Attach]</td>
</tr>
<tr>
<td>REST Service Object</td>
<td>RestMethod</td>
</tr>
<tr>
<td></td>
<td>RestURI</td>
</tr>
<tr>
<td></td>
<td>ProducedMediaType, [ConsumedMediaType]</td>
</tr>
<tr>
<td></td>
<td>[Query Parameters]</td>
</tr>
<tr>
<td></td>
<td>[Path Parameters]</td>
</tr>
<tr>
<td></td>
<td>[Matrix Parameters]</td>
</tr>
<tr>
<td></td>
<td>[JSON Parameters]</td>
</tr>
<tr>
<td></td>
<td>[XML Parameters]</td>
</tr>
</tbody>
</table>

Table 4.6 | Action Framework Command Parameters Categorised by Service

Table 4.6 shows the attributes that the Action Framework Commands need to have to implement a specific service successfully. The attributes in “[]” are optional and specific to the command implementation. We have already explained what each of these attributes means in section 4.4.3 and 4.4.4. Refer to Appendix B.3 for an example of a Framework Command that implements the RESTful Service.
The translation from Service Object response to Framework Command will be explained in section 4.5.5.

A Framework Command will be defined for each REST service that the system needs to execute, by reusing the REST Service Object. The same applies for other two services. The reusability of the Service Objects is straightforward as they have generic nature. Moreover, an Action Framework Command is a distinct component that can be reused in many Actions. For Example, the Framework Command that implements the Mail Service Object can be reused by an Action that needs to attach a document in the mail, another Action that needs to send a plain text, and a third Action that needs to send HTML content.

### 4.5.4 Hybrid Framework Command

There are cases where a flow needs to trigger other flows. To address these cases, the action is allowed to define and execute a Framework Command that implements a Flow Service. Therefore, an Action Framework Command is a Triggering Framework Command at the same time. This command is called Hybrid Framework Command. As an Action Framework Command, there is no need for an adapter to do any messaging translation from the triggering event to a Framework Command. On the other hand, as a Triggering Framework Command, the command execution returns the standard response defined in previous section 4.5.2, which is already a Framework Command. So, there is no need for any translation from Service Object response to Framework Command.

The response in this case is a status on flow initiation. Therefore, the current workflow execution is not waiting for any feedback from the execution of the initiated Flows. This is the mechanism that File Reader uses to trigger flows.

### 4.5.5 Translation of Service Object response into a Framework Command

Flow Service Object response is already a Framework Command explained and displayed on section 4.5.2. While, Service Objects of Action Framework Command return different message responses that need to be translated. This interface incompatibility is handled by Action Framework Command Adapters illustrated in Table 4.7
### Action Framework Command Adapters

<table>
<thead>
<tr>
<th>Action Framework Command Adapters</th>
<th>Service Object Response</th>
<th>Framework Command Attributes</th>
</tr>
</thead>
<tbody>
<tr>
<td>MailResponseAdapater</td>
<td>Delivery Status</td>
<td>provide Delivery Status as an attribute</td>
</tr>
<tr>
<td>DatabasetResponseAdapater</td>
<td>Result Set</td>
<td>extract column names and values from Result Set</td>
</tr>
<tr>
<td>RESTResponseAdapater</td>
<td>XML Response</td>
<td>extract attributes using XPath</td>
</tr>
<tr>
<td></td>
<td>plain text</td>
<td>provide response text as an attribute</td>
</tr>
</tbody>
</table>

Table 4.7| Action Framework Command Adapters: extraction of Framework Command Attributes from Service Object responses

#### I. Translation of a Mail Service Object response is straightforward. The *Delivery Status* is added as a command attribute, having so the below response structure:

\[
\text{AOW} | \text{MAIL}\text{|Request Command Name | Command ID | Attributes} [ \begin{cases} \text{CommandResult: \{Success, Fail\}}, \\
\text{ResultMessage :\{ Success, Fail\}}, \\
\text{Delivery Status :\{ status returned from the Service Object\}} \end{cases} ]
\]

#### II. Translation of a Database Service Object response is not a complicated task as soon as the system defines the constraint that every store procedure that this service executes returns a cursor. The adapter iterates all this tabular structure by extracting the column names and values. If for a specific column there is more than one row, then the values are stored in an array list of Objects. Response Structure:

\[
\text{AOW} | \text{DATABASE}\text{|Request Command Name | Command ID | Attributes} [ \begin{cases} \text{CommandResult: \{Success, Fail\}}, \\
\text{ResultMessage :\{ one of Result Message List values\}}, \\
\text{Column Name :\{ Column Value\}} \end{cases} ]
\]

Appendix B.4 displays for the Message List possible values.

#### III. REST Service Object response translation was a challenge as:

- The responses are dependent on the service that is being accessed.
- The structure of the response differs from one service to another. There are many formats available like plain text, XML and JSON.

So, it is not possible to define a pattern that is applied to all cases. The parsing of the response and extraction of all its information into a Framework Command was simplified by:

- Supporting only plain text and XML responses.
- It parses the response to extract only the information needed for the workflow logic by using Dom Library and XPath.
In this case, the user needs to define the response in the data model by configuring the parameters with default value the related XPath. Response Structure:

```
<table>
<thead>
<tr>
<th>AOW</th>
<th>REST</th>
<th>Request Command Name</th>
<th>Command ID</th>
<th>Attributes</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>CommandResult: [Success, Fail],</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>ResultMessage : { Response Error Code},</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>Parameter Name : { XPaths extracted value}]</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
```

For example for `GetSubsriberData` response:

```
<GetSubsriberData>
  <balance>5</balance>
  <class_of_Service_Code>10002</class_of_Service_Code>
  <discount_IDs>
    <discount_ID>702</discount_ID>
    <endDate>2016-09-20T00:00:01:00</endDate>
    <startDate>2016-08-21T00:00:01:00</startDate>
  </discount_IDs>
  <discount_ID>701</discount_ID>
  <endDate>2016-08-20T00:00:01:00</endDate>
  <startDate>2016-08-21T00:00:01:00</startDate>
  <feature_Package_IDs>
    <endDate>2016-09-20T00:00:01:00</endDate>
    <feature_Package_ID>Monthly_Bundle</feature_Package_ID>
    <startDate>2016-08-21T00:00:01:00</startDate>
  </feature_Package_IDs>
  <msisdn>447824440412</msisdn>
  <sim_State_Code>1</sim_State_Code>
</GetSubsriberData>
```

To retrieve the needed information, the below configurations are defined by the response parameters:

<table>
<thead>
<tr>
<th>Parameter Name</th>
<th>Default Value (XPATH)</th>
</tr>
</thead>
<tbody>
<tr>
<td>CLASS_OF_SERVICE_CODE</td>
<td>//Class_Of_Service_Code</td>
</tr>
<tr>
<td>BALANCE</td>
<td>//balance</td>
</tr>
<tr>
<td>SIM_STATE_CODE</td>
<td>//sim_State_Code</td>
</tr>
<tr>
<td>BUNDLES</td>
<td>GetSubsriberData/feature_Package_IDs/feature_Package_ID</td>
</tr>
<tr>
<td>BUNDLES_START_DATE</td>
<td>GetSubsriberData/feature_Package_IDs/startDate</td>
</tr>
<tr>
<td>BUNDLES_END_DATE</td>
<td>GetSubsriberData/feature_Package_IDs/endDate</td>
</tr>
<tr>
<td>BUCKETS_START_DATE</td>
<td>GetSubsriberData/discount_IDs/startDate</td>
</tr>
<tr>
<td>BUCKETS_END_DATE</td>
<td>GetSubsriberData/discount_IDs/endDate</td>
</tr>
<tr>
<td>BUCKETS</td>
<td>GetSubsriberData/discount_IDs/discount_ID</td>
</tr>
</tbody>
</table>

Table 4.8] REST Response Parameter Configurations

Based on these mapping the `GetSubsriberData` response is translated into the below Framework Command:
Obviously, this design introduces system limitations as this translation might not work properly when both below conditions are met:

- The XML response represents complicated objects
- The workflow needs the whole information.

AOW offers capabilities to handle these situations:

- If these cases are identified when the system is adapted to a predefined environment. The manipulator classes can be introduced to perform specific logic.
- Otherwise, the user has to do more advanced configurations by introducing Bean Shell scripts to manipulate the response.

4.6 RESTful Web Server

RESTful Web Server is part of the Communication Layer. This component enables restful triggers. It provides the flow triggering service via REST. Thus, other systems can call this service to trigger flows inside the framework. It has the below characteristics:

- It exposes GET service
- It produces XML response
- URI: http://localhost:8080/ActionOrientedFramework/aow/flow/
- First Parameter : @PathParam("CommandName")
- One or more Query Parameters that will be translated to Command Parameters
- It uses JAXBElement to translate Framework Command response into XML response.
The operations of this service are:

- Translate request to Triggering Framework Command
- Execute Framework Command

Find an example of REST Server request/response in Appendix B.5.

4.7 File Reader

File Reader is one of the triggering components of AOW. It is useful if the systems in the enterprise environment logg events in files where a row represents an event. Different files from various resources can be merged into AOW files following a predefined structure. This process is domain specific and should be handled outside the framework. Once the file repository is populated the file reader process reads the files and identifies the Flows that should be initiated based on preconfigured filters.

![File Reader Workflow](image-url)
This process is implemented using the workflow mechanism. It is the first workflow defined, and it is considered a System Workflow. Figure 4.5 visualises the File Reader workflow as a flowchart. There are three actions identified from this process detailed in Table 4.9.

<table>
<thead>
<tr>
<th>Action</th>
<th>Description</th>
<th>Executes</th>
<th>In Parameters</th>
<th>Out Parameters</th>
</tr>
</thead>
<tbody>
<tr>
<td>LoadFile</td>
<td>Reads the whole file and loads it in InMemoryFile Object. After that removes the file to archives</td>
<td>Manipulator Class</td>
<td>InMemoryFile RowNumber</td>
<td></td>
</tr>
<tr>
<td>ReadRow</td>
<td>Reads the row and checks which flow should be triggered</td>
<td>Framework Command</td>
<td>CurrentRow RowNumber</td>
<td>CurrentRow LoopFactor FlowNames LoopFactor</td>
</tr>
<tr>
<td>TriggerFileBasedFlows</td>
<td>Trigger the flows defined by the previous action.</td>
<td>Hybrid Framework Command</td>
<td>FlowNames</td>
<td></td>
</tr>
</tbody>
</table>

Table 4.9 | File Reader Workflow Actions

The first action which is attached to FileReaded flow is LoadFile. This action reads the whole file, loads it in InMemoryFile object and moves the file to archives. The file is loaded in memory as the process loops by processing each file record. In this way are avoided many I/O communications. This action executes only a manipulator class to perform these operations. It outputs InMemoryFile and RowNumber (number of rows in the file).

Next, if the file is not empty ReadRow action is executed. It reads the row and checks the file filters to identify possible Flows that should be triggered by the event represented in this row. These flows are stored in FlowNames separated by commas. It also calculates the LoopFactor based on CurrentRow and RowNumber. These operations are implemented in a stored procedure called by a Framework Command (Database Service).

If LoopFactor is not zero, then the TriggerFileBasedFlows is picked. It configures a Hybrid Framework Command. This command triggers all the flows defined by the previous action.

The rules illustrated with rhombus in Figure 4.5 are translated as next action rules. Table 4.10 illustrates this translation.

<table>
<thead>
<tr>
<th>Flow</th>
<th>Current Action</th>
<th>Next Action</th>
<th>Flow Parameter</th>
<th>Pattern</th>
<th>Rule Group</th>
</tr>
</thead>
<tbody>
<tr>
<td>FileReader</td>
<td>LoadFile</td>
<td>ReadRow</td>
<td>InMemoryFile</td>
<td>^[^0]</td>
<td>1</td>
</tr>
<tr>
<td>FileReader</td>
<td>ReadRow</td>
<td>ReadRow</td>
<td>InMemoryFile</td>
<td>^[0]$</td>
<td>2</td>
</tr>
<tr>
<td>FileReader</td>
<td>ReadRow</td>
<td>ReadRow</td>
<td>LoopFactor</td>
<td>^continue$</td>
<td>2</td>
</tr>
<tr>
<td>FileReader</td>
<td>ReadRow</td>
<td>TriggerFileBasedFlows</td>
<td>FlowNames</td>
<td>^[^0]</td>
<td>3</td>
</tr>
<tr>
<td>FileReader</td>
<td>TriggerFileBasedFlows</td>
<td>ReadRow</td>
<td>LoopFactor</td>
<td>^continue$</td>
<td>4</td>
</tr>
</tbody>
</table>

Table 4.10 | File Reader Next Action Rules
The workflow selection logic uses predefined configurations of the File Structure and File Filters. Refer to Appendix B.6 for details and examples of these configurations.

4.8 Logging and Monitoring

AOW implements a Logging framework for debugging and monitoring purpose. There are three types of logs supported by the framework:

- **System**: logs framework related information like: framework start up and shut down activates, system statuses, thread pooling, and database connection pooling.
- **Transaction**: logs the whole process from triggering command to workflow execution
- **Debug Log**: logs the same process as transaction log but in more details.

Each row has an indication that informs the importance and impact of the information logged as per below:

```
INFO=5;
OK=4;
WARNING=3;
MAJOR=2;
CRITICAL=1;
```

Each row has a log level from 1 to 100. The system logs only the rows with log level from 0 to n, where n is a value defined in system configurations.

**System Logs** are used to monitor the system if:

- system reboots
- system fails
- maximum thread pool size reached
- maximum database pool size is reached
- performance issues identified
- security issues are identified
- Any strange system behaviour identified

**Transaction Logs** are used to monitor Workflow Initiation and Workflow Execution phases for each triggering event and each flow.

**Debug Logs** are monitored to check for more details the Workflow Initiation and Workflow Execution phases for each triggering event and each flow. These logs should not be activated all the time as they consume considerable storage, especially if these logs are stored on the
same server with the application. For this reason, debug logging can be activated for specific flows or commands using log level. Thus, it might be activated for new flows that need to be monitored or existing flows that might not be executed as expected. Once the investigation is finished, the debugging for these flows should be deactivated.

All the three logs are structured. For example, transaction log and debug log starts with the Flow Id (logging Workflow Execution phases), or Framework Command Id (logging Workflow Initiation phase).

Example of transaction logs:

*Flow@1488843337|Action@63759356|{List_Id=1}|Flow.initialiseparameters,20160821144953.366, MESSAGE(flow params updated from action params),

*FrameworkCommand@1488847337|PulseLogic.REQUEST,AOW,FLOW,Pulse,000687507223,20160821144951.504,TIME_MILIS:1471787391500,TIME:20160821144951.500 Sun,

Having a specific structure make these logs useful meaning that they can be loaded in a Data warehouse schema and used for BI, Data Mining and Reconciliation process. For the moment AOW does not offer a visual system for workflow monitoring. Therefore, logging is the only available monitoring capability for the framework.

**4.9 System and Workflow Deployment**

When the system is firstly integrated into a specific enterprise environment, it needs to be adapted. Below steps should be performed:

- Setting up the integrations and Talend Jobs to translate other protocols to REST and vice versa.
- Configure the Web Server and System Properties
- Adding Manipulator classes for the identified cases that need input/output manipulation (especially for REST services)
- Configure the File Filters and File Structure

After these steps, the application should be deployed in all nodes of Application/ Web Server. A replicated database model should be configured on all database nodes.
The system is redeployed only in case of changes like:

- Adding new Manipulators
- Adding new Integration Services
- System Upgrades

In all these cases system restart is required. Restarting a middleware in an enterprise environment is a dangerous action that should be performed in low traffic hours as it impacts the services. AOW implements a safe shut down process that follows the below steps:

- The node is isolated from triggering requests
- The system waits until the flow queue is empty
- The system waits until all the running flows have finished the execution normally.
  It checks the list of flows that are being executed which is maintained by the Flow Executer Manager.

In case of system upgrades, the already configured workflows (Flow, Actions, Framework Commands, Fick Flow and Next Action rules) need to be migrated from the old system to the new one. The changes of the upgraded system should not impact the logic and configurations of the existing workflows. These can not be ensured as it depends on the needs. Therefore, the system should provide a way to manage different versions of workflows. For the moment AOW does not have this capability. Therefore, it does not support workflow versioning.

*Workflow deployment* is another process that needs to be defined. There should be at least two environments:

- Lab Environment: the workflows are developed and tested
- Production Environment: the workflows are deployed and executed

It is crucial to have these two distinct environments to ensure the quality of service and avoid issues. In addition, the modification of a service does not impact the current execution.

The current workflow deployment process is managed in database level by means of changed flags. When deploying in production new workflows, there is no need to stop any service or running workflows. Deploying changes to existing workflows can be performed in two ways:

- The impacted flow should be deactivated and the running flows should normally be terminated. After that, the changes can be deployed. If this procedure is followed
correctly the service is not impacted as the deactivated flows are added back to the flow queue (explained in section 4.3.1).

- The changes can be deployed without deactivating any flow. It might be dangerous as the changes take effect for the current flow executions as well. Therefore, it is not a recommended procedure.

4.10 Summary

This chapter provides AOW implementation details. It is observed that the technology choices are driven by the lightweight criteria. Service generalisation is performed by introducing configurations in Framework Command level. This is an important feature that enables service reusability. The three types of services are explained along with their corresponding implementations in Framework Commands. The Framework Commands operate as a standard inside AOW, meaning that the triggers and the service responses are translated to these objects by means of an adapter.

RESTful web server exposes the Flow Service of the framework to other systems. This enables the external systems to trigger workflows inside the framework. File Reader module is a critical component implemented using the workflow modelling and execution capabilities of the framework. The monitoring of workflow and system execution is performed by means of three types of logs, system, transaction, and debug logs.
Chapter 5. Evaluation

5.1 Chapter Overview

This chapter demonstrates the evaluation of the project against its aim and objectives. In order to perform this evaluation the procedure below is followed:

- The evaluation scenarios are simulated in a reflection of a real world Telecom environment. The scenarios are real marketing requests that will be modelled and executed using the AOW framework. This demonstration is used to evaluate the pilot functionality.
- Three different types of test suites are analysed to evaluate system accuracy and correctness. Unit testing and code coverage reports are demonstrated along with White Box and Black Box testing.
- Centralisation is one of the motivations of this project. It is assessed by demonstrating the SOA architecture of the system as well as business process modelling and execution centralisation.
- The evaluation of lightweight performed against its definition in the context of this project is provided in section 1.2. This evaluation is based on the technology choices and design decisions discussed in Chapter 3.
- Reusability is one of the most important goals of this project. Two levels of reusability are evaluated (Service and Action levels). It is performed based on the modelling of 6 evaluation scenarios. The calculation of reusability is focused on the number of reusable Actions and Framework Commands designed to model these scenarios.

5.2 Evaluation of Workflow Modelling and Execution

AOW’s main scope is to provide a tool for BPM in the enterprise. The functionality evaluation is based on scenarios simulation using the AOW framework. It demonstrates that AOW offers the means to model these business processes in workflows and execute them as expected. The construction scenario was used during the design and implementation of the system. This section focuses on the evaluation scenarios consisting of CBS scenarios and Loyalty scenario. First, a replication of a telecommunication environment is set up, based on which the scenarios are modelled and executed.
5.2.1 Reflection of real world scenarios

The best approach to assess this project would have been the implementation of these scenarios in a real industry environment, where the system had to be adapted to unanticipated situations and used by independent developers. However, this was not followed as:

- AOW is a pilot, not a completed product.
- There were time limit constraints.
- It exposes a risk of shifting from the real concerns of the project (Centralisation, Lightweight and Reusability) to only testing the product.

Therefore, the evaluation of the project is performed based on simulated scenarios using real marketing requests. The evaluation scenarios discussed in section 3.3 are in the domain of telecommunication. Based on the author experience in this domain, the simulation of these scenarios is a reflection of the real telecommunication environment. This reflection is illustrated in the figure below:

![Diagram of AOW in a mirror of Telecom Environment]

Figure 5.1 Adaption of AOW in a mirror of Telecom Environment.

We have already defined the terminologies in the introduction of this project. Table 5.1 shows the systems that have been implemented and what they do. The services that these systems provide are implemented in REST. Therefore, Talend ESB has not been used as there was not any need for protocol translation.
### System Description

<table>
<thead>
<tr>
<th>System</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Electronic Top Up Application</td>
<td>A simple application that gets as input the MSISDN and the Recharge Amount and increases the balance in Billing system by using the corresponding service.</td>
</tr>
<tr>
<td>Billing</td>
<td>Maintains a simple customer profile with details on promotions like Bundles and Buckets. It exposes the services to get the profile information and to modify it.</td>
</tr>
<tr>
<td>CRM</td>
<td>Maintains the segments of customers (Lists of MSISDNs). It exposes the service to check the customer segment.</td>
</tr>
<tr>
<td>SMS Centre</td>
<td>Uses an existing SMS Gateway to Send and Receive SMS.</td>
</tr>
</tbody>
</table>

Table 5.1: Reflection of Telecommunication Environment. Systems that have been implemented to support the scenarios simulation.

### 5.2.2 Customer Base Segmentation scenarios

The first group of evaluation scenarios are marketing requests in CBS. These scenarios are similar to the construction scenario and are defined in the table below:

<table>
<thead>
<tr>
<th>Scenario No</th>
<th>Scenario Name</th>
<th>Marketing Request Specifications</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Sample Acquisition</td>
<td>If the customer performs the First Call and is part of “Sample Acquisition” segment, then s/he will get a Monthly Bundle with (100 minutes and 500 MB) valid for one month. The customer is eligible to get this promotion only once per month. The customer should get a notification SMS after the successful bundle subscription.</td>
</tr>
<tr>
<td>2</td>
<td>Professionals Promotion</td>
<td>If the customer buys a bundle (Bundle Subscription) and is part of “Everyday Professional” segment, then s/he will get a Bucket with 100 minutes (Voice Bucket) valid for one Month. The customer can make use of this promotion up to 2 times per month. The customer should get a notification SMS after the successful bundle subscription.</td>
</tr>
<tr>
<td>3</td>
<td>Technology fans</td>
<td>If the Data Bundle of the customer expires (Bundle Expiration) and is part of “Technology Fans” segment, then s/he will get a Bucket with 100 MB (Data Bucket) valid for one Month. The customer might benefit this promotion up to 3 times per month. The customer should get a notification SMS after the successful bundle subscription.</td>
</tr>
</tbody>
</table>
If the customer performs a *Balance Top Up* with **more than 5 Pounds and less than 10 Pounds** and is part of “University Students” segment, then s/he will get a *Weekly Bundle* with (100 minutes and 500 MB) valid for one week. The customer gets this promotion *in each recharge*. The customer should **get a notification SMS** after the successful bundle subscription.

If the customer performs a *Balance Top Up* with **10 Pounds or more**, and is part of “University Students” segment, then s/he will get a *Mega Bundle* with (200 minutes and 1 GB) valid for one Month. The customer receives this promotion **only once per month**. The customer should **get a notification SMS** after the successful bundle subscription.

This information is extracted by the marketing request specifications, and it is synthesised in Table 5.3.
Normally each of these scenarios should be modelled in a separate workflow, but the general pattern illustrated in Table 5.3 has been identified. The construction scenario follows the same pattern. Therefore, the workflow that models the construction scenario (Figure 3.2) models all these scenarios as well.

In the tables below is provided a summary of Actions and Framework Command configurations. Refer to Appendix A for Full Data Model Configurations.

<table>
<thead>
<tr>
<th>Actions</th>
<th>Description</th>
<th>Framework Command</th>
<th>Service</th>
<th>Executed in System</th>
</tr>
</thead>
<tbody>
<tr>
<td>Get Segment</td>
<td>Checks in CRM if the MSISDN is in any Segment</td>
<td>Check Segment</td>
<td>REST Service</td>
<td>CRM</td>
</tr>
<tr>
<td>Get Subscriber Data</td>
<td>Query the Profile of the MSISDN in Billing</td>
<td>Get Customer Profile</td>
<td>REST Service</td>
<td>Billing</td>
</tr>
<tr>
<td>Get Promotion Data</td>
<td>Executes get Promo Data procedure in Database, it checks promotion conditions based on the segment that the customer is</td>
<td>Get Promotion Data</td>
<td>Database Service</td>
<td>Database</td>
</tr>
<tr>
<td>Subscribe Bundle</td>
<td>Subscribes the Bundle in Billing profile</td>
<td>Subscribe Bundle</td>
<td>REST Service</td>
<td>Billing</td>
</tr>
<tr>
<td>Remove Bundle</td>
<td>Removes the Bundle from customers profile in Billing</td>
<td>Remove Bundle</td>
<td>REST Service</td>
<td>Billing</td>
</tr>
<tr>
<td>Subscribe Bucket</td>
<td>Subscribes the Bucket in Billing profile</td>
<td>Subscribe Bucket</td>
<td>REST Service</td>
<td>Billing</td>
</tr>
<tr>
<td>Remove Bucket</td>
<td>Removes the Bucket from customers profile in Billing</td>
<td>Remove Bucket</td>
<td>REST Service</td>
<td>Billing</td>
</tr>
<tr>
<td>Store Transition</td>
<td>Store Bonus subscription as a transaction in Database, These serve to check promotion frequency</td>
<td>Store Transition</td>
<td>Database Service</td>
<td>Database</td>
</tr>
<tr>
<td>Send SMS</td>
<td>Sends the notification SMS to the customer</td>
<td>Send SMS</td>
<td>REST Service</td>
<td>SMS Centre</td>
</tr>
</tbody>
</table>

Table 5.4 Components Model Configuration Summary

All the operations defined on the workflow in Figure 3.2 are mapped to an Action that executes a corresponding Framework Command. We can notice that only REST and Database services were enough to serve these scenarios operations. In Appendix A are displayed all configurations like Action input and output parameters, and Framework Parameters. In addition, Flow Parameter – Action Parameter Mappings are defined.

Workflow Execution: Once a trigger is identified from the File Reader the workflow is added to the queue. After that, the manager initiates the workflow execution that starts with the execution of Get Segment Action. The execution follows the flow defined in Figure 3.2. This flow is executed correctly for all the scenarios. Debug Logging is used to evaluate the execution where each execution step is recorded along with the data (parameters). In
addition, after each execution the customer receives the confirmation SMS and the bonus is applied to his profile in Billing.

5.2.3 Loyalty scenario

The customers gather points from different events like First Call, Bundle Subscriptions, and Balance Top Ups. The points can be redeemed to get different bonuses (bundles, vouchers, Mobile Devices). There are three workflows identified in this request:

- Point Rewording Workflow adds loyalty points on customer events
- Birthday Workflow adds loyalty points on customer birthday
- Point Redemption Workflow readmes points on customer request.

Figure 5.2| Loyalty Program, Point Rewarding Workflows

Figure 5.2 illustrates the two workflows that add Loyalty Points. Each operation is translated into an action shown in the below table:
<table>
<thead>
<tr>
<th>Flow</th>
<th>Actions</th>
<th>Description</th>
<th>Framework Command</th>
<th>Service</th>
<th>Executed in System</th>
</tr>
</thead>
<tbody>
<tr>
<td>Point Rewarding Workflow</td>
<td>Get Subscriber Data</td>
<td>Query the profile of the MSISDN in Billing</td>
<td>Get Customer Profile</td>
<td>REST Service</td>
<td>Billing</td>
</tr>
<tr>
<td>Point Rewarding Workflow</td>
<td>Add Points</td>
<td>Calculates the points based on the event and customer profile data</td>
<td>Add Loyalty Points</td>
<td>Database Service</td>
<td>Database</td>
</tr>
<tr>
<td>Point Rewarding Workflow</td>
<td>Send SMS</td>
<td>Sends the notification SMS to the customer</td>
<td>Send SMS</td>
<td>REST Service</td>
<td>SMS Centre</td>
</tr>
<tr>
<td>Birthday Workflow</td>
<td>Get Birth Date MSISDNs</td>
<td>Gets from CRM the list of customers that have Birthday</td>
<td>Check Segment</td>
<td>REST Service</td>
<td>CRM</td>
</tr>
<tr>
<td>Birthday Workflow</td>
<td>Get Customers Data</td>
<td>Loop the MSISDNs list and get the data for the current MSISDN. Controls the loop</td>
<td>Get Customer Profile</td>
<td>REST Service</td>
<td>Billing</td>
</tr>
<tr>
<td>Birthday Workflow</td>
<td>Add Points</td>
<td>Calculates the points based on the event and customer profile data</td>
<td>Add Loyalty Points</td>
<td>Database Service</td>
<td>Database</td>
</tr>
<tr>
<td>Birthday Workflow</td>
<td>Send SMS</td>
<td>Sends the notification SMS to the customer</td>
<td>Send SMS</td>
<td>REST Service</td>
<td>SMS Centre</td>
</tr>
</tbody>
</table>

Table 5.5] Loyalty Program, Point Rewarding and Birthday workflow Components Model Configuration Synthesis

Point Redemption Workflow is illustrated in Figure 5.3.

Figure 5.3] Loyalty Program, Point Redemption Workflow
The table below provides all the operations used in this workflow. Again, each of these transactions is translated into one single action, which in turn executes a Framework Command.

<table>
<thead>
<tr>
<th>Actions</th>
<th>Description</th>
<th>Framework Command</th>
<th>Service</th>
<th>Executed in System</th>
</tr>
</thead>
<tbody>
<tr>
<td>Check Bonus Threshold</td>
<td>Checks if the customer has enough points to get</td>
<td>Check Bonus Threshold</td>
<td>Database Service</td>
<td>Database</td>
</tr>
<tr>
<td></td>
<td>the requested bonus</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Get Subscriber Data</td>
<td>Query the Profile of the MSISDN in Billing</td>
<td>Get Customer Profile</td>
<td>REST Service</td>
<td>Billing</td>
</tr>
<tr>
<td>Subscribe Bundle</td>
<td>Subscribes the Bundle in Billing profile</td>
<td>Subscribe Bundle</td>
<td>REST Service</td>
<td>Billing</td>
</tr>
<tr>
<td>Remove Bundle</td>
<td>Removes the Bundle from customers profile in Billing</td>
<td>Remove Bundle</td>
<td>REST Service</td>
<td>Billing</td>
</tr>
<tr>
<td>Subscribe Bucket</td>
<td>Subscribes the Bucket in Billing profile</td>
<td>Subscribe Bucket</td>
<td>REST Service</td>
<td>Billing</td>
</tr>
<tr>
<td>Remove Bucket</td>
<td>Removes the Bucket from customers profile in Billing</td>
<td>Remove Bucket</td>
<td>REST Service</td>
<td>Billing</td>
</tr>
<tr>
<td>Send EMail</td>
<td>Send to Warehouse an email with the order details to ship the device to the customer.</td>
<td>Send Email</td>
<td>EMAIL Service</td>
<td>EMAIL Server</td>
</tr>
<tr>
<td>Send SMS</td>
<td>Sends the notification SMS to the customer</td>
<td>Send SMS</td>
<td>REST Service</td>
<td>SMS Centre</td>
</tr>
</tbody>
</table>

Table 5.6] Loyalty Program, Point Redemption workflow Components Model Configuration Synthesis

The rhombuses in the workflows in Figure 5.2 and Figure 5.3 are translated into Next Action Rules. The workflow execution is performed as illustrated in these figures. As already stated, the debug logging is used to ensure that this execution is carried out as expected in all the steps.

5.2.4 Analysis on workflow modelling and execution evaluation

In the two previous sections, it was demonstrated how 6 different evaluation scenarios are modelled into AOW workflow Data Model and how they are executed. The analysis of this demonstration concludes in the following evaluation results:

- AOW framework provides the tools to model the workflows by means of AOW Data Model.
  - The Environment Triggers (REST, File Reader and Pulse) covered all the triggers of these scenarios.
  - All the operations of the workflows were modelled in Actions which were executing a Framework Command.
  - The services (REST Service, Database Service, and Email Service) covered all the possible operations that these scenarios needed.
The business rules were translated in Pick Flow and Next Action Rules.

- AOW Workflow engine executed these workflows as expected. During the workflows executions debug logging was active and was recording every step in the execution process.
  - Debug logging is used to check the control flow of the workflows execution. It has investigated every step of workflow initiation and workflow execution phases. This execution is verified against the workflow diagrams defined for each scenario.
  - Debug logging is used to check the data flow during the workflow execution. It records the parameters before and after each step. Parameter initialisation is checked against the parameter mapping defined in the Data Model.
  - The workflow execution is checked against the expected results. For example one of the expected results of Sample Acquisition scenario is: the customer should have the bonus reflected on his customer profile and should have received the SMS notification.

**Conclusion:** The simulation of 6 evaluation scenarios demonstrated that the AOW framework provides the capabilities to model and execute enterprise business processes in Telecommunication Marketing Requests.

### 5.3 AOW framework Testing

Two of the major main features of AOW framework deemed to be assessed are correctness (freedom from faults) and accuracy (freedom from errors). There are three types of test suites conducted in order to evaluate these important system qualities:

- Unit Tests and code coverage
- White Box Testing
- Black Box Testing.

JUnit Framework, which is the default tool for Unit tests in Java, is used to implement the Unit Test Suite. They were performed in parallel with the system implementation. They were used for regression testing as well, in the case of changes in design or refactoring. The Unit Test Suite is executed with zero errors.

![Figure 5.4 | Run Result of Unit Test Suite](image-url)
Code coverage is a report that represents how many lines of code are executed and tested when the Unit Test suite is executed. The higher the code coverage is, the higher the level of correctness and accuracy is.

EclEmma Eclipse plug-in is used to run the code coverage. Figure 5.5 illustrates the code coverage report that is detailed in package level. *The total code coverage achieved is 83.8%.* The important packages of the framework have code coverage up to 100%. The packages such as logging, tools and manipulators have significant low code coverage, but these are simple classes mainly evaluated with black box testing.

![Image of code coverage report](image)

**Figure 5.5| AOW Framework Code Coverage report**

White box testing covers test cases are categorised as follows:

- Workflow Triggering, Pick Flow
- Flow execution
- Action execution
- Framework Command Execution
- Parameter Initialisation
- Next Action Algorithm
- File Reader

The debug log was used to check the execution against the expected results. Figure bellow illustrates an example of a Pick Flow test case. These tests resulted useful to identify faults in implementation, bugs and configuration errors that were constantly corrected up to the point
that all the tests were successful. These tests were performed using both construction and evaluation scenarios. Due to time constraints, not all the possible test scenarios were covered, but only the most important cases related to each category listed above.

<table>
<thead>
<tr>
<th>Test ID</th>
<th>Test Name</th>
<th>Prerequisites</th>
<th>Expected Results</th>
<th>Log ID</th>
<th>Results Description (Logs)</th>
<th>result</th>
</tr>
</thead>
<tbody>
<tr>
<td>PK_Flow_03</td>
<td>Pick the right flow</td>
<td>Init flow conditions: MSISDN = &quot;0777&quot;, BONUS_NAME=&quot;Monthly&quot;</td>
<td>Both flows are selected</td>
<td>DebugLog_20160721</td>
<td>,CommandArrived,AOW,AOW,FLOW,RegisterSubscriberBonus,-00522610129,20160712160920.411,</td>
<td>Passed</td>
</tr>
<tr>
<td></td>
<td></td>
<td>init CMD actual values: &quot;MSISDN&quot; = &quot;07771413242&quot;, &quot;BONUS_NAME&quot; = &quot;MonthlyBundle&quot;</td>
<td></td>
<td></td>
<td>MSISDN=07771413242,BONUS_NAME=&quot;MonthlyBundle&quot;,</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>Group 1 has 4 flows: SubscribeBonus, SubscribeBonusLoyalty</td>
<td></td>
<td></td>
<td>4,AOW,20160712160920.897,FlowsAreSelected,RegisterSubscriberBonus,-00522610129,</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>Flow Profile: no configs for these flows</td>
<td></td>
<td></td>
<td>[FLOW_1=SubscribeBonus,LOYALTY, FLOW_0=SubscribeBonus],</td>
<td></td>
</tr>
</tbody>
</table>

Figure 5.6 | White Box Testing, Pick Flow test case scenario example

Black box testing was performed during evaluation scenarios execution. For each of the scenarios the expected results were checked, like the bonus on customer profile, receipt of the SMS or receipt of the Email. All these tests resulted successfully.

Conclusion: The accuracy and correctness of the system are evaluated to an acceptable level considering that all Unit Tests resulted successfully with adequate code coverage of 83.8%. In addition, it is supported by the successful white box and black box testing covering all the important test cases. To improve this evaluation even further, the code coverage should be increased to 100% and white box test cases should include all the test scenarios.

5.4 Evaluation of Centralisation

Centralisation meaning and importance has been already discussed in section 1.1. Centralisation evaluation is performed based on two criteria:

- **AOW architecture**: It operates in an SOA architecture offering and consuming web services. By these means, AOW framework can be integrated with other systems in the enterprise to execute the operations of the business processes remotely. This system capability has been already proved with the simulation of scenarios. In section 5.2.1 is illustrated the environment that AOW was integrated to simulate the 6 evaluation scenarios. In addition, the scenario modelling defines clearly that the operations of the processes are performed remotely. Therefore, it is demonstrated that AOW is operating in an SOA environment.
- **AOW functionality**: AOW offers a single tool to implement and execute all the business processes. In section 5.2 it is demonstrated that the framework possesses this capability. All 6 evaluations scenarios are modelled using AOW Data Model and are executed by AOW Workflow Engine. Therefore, there is one single repository to store and one unique mechanism to run the business processes, simplifying so the maintenance and monitoring of these processes.

**Conclusion**: Based on its SOA architecture and its functionality to offer a single tool for business processes modelling and execution, AOW is a centralised solution.

### 5.5 Evaluation of Lightweight

One of the aims of this project is to create a lightweight BPM tool for the enterprise. AOW framework is evaluated against three requirements of lightweight as discussed in section 1.2.

- **The system does not require sophisticated application architecture.** It is a composition of lightweight technologies.

<table>
<thead>
<tr>
<th>Technology</th>
<th>Used for AOW</th>
<th>Lightweight characteristic</th>
</tr>
</thead>
<tbody>
<tr>
<td>RESTful Web Services</td>
<td>Integration Layer</td>
<td>REST is a lightweight protocol compared to its alternative SOAP. Analysis performed in section 2.2.3</td>
</tr>
<tr>
<td>Talend ESB</td>
<td>Integration Support</td>
<td>Talend ESB is a lightweight open source technology compared to other open source or commercial ESB solutions. Analysis performed in section 2.2.5</td>
</tr>
<tr>
<td>Java EE</td>
<td>Core Application</td>
<td>Java is a lightweight technology. It is an open source, platform-independent programming language. Discussed in section 4.2</td>
</tr>
<tr>
<td>Apache Tomcat</td>
<td>Application Server</td>
<td>It is an open source application server known for its lightweight and highly adaptable architecture</td>
</tr>
<tr>
<td>Linked Blocking Queue</td>
<td>Flow Queuing</td>
<td>It is a straightforward and lightweight queue option compared to other messaging technologies that work like middleware. Analysis performed in section 3.10.1</td>
</tr>
<tr>
<td>Bean Shell Scripting</td>
<td>Action parameter manipulations</td>
<td>It is a Java lightweight scripting mechanism. Discussed in section 4.3.3</td>
</tr>
<tr>
<td>Thread Pooling</td>
<td>Multithreading</td>
<td>It is a lightweight mechanism as it improves thread allocation, resulting in better resource management. Analysed in section 3.10.2</td>
</tr>
<tr>
<td>Database Connection pooling</td>
<td>Database Connections</td>
<td>It is a lightweight mechanism as it improves connection creation and allocation, resulting in better resource management. Analysed in section 3.10.5</td>
</tr>
</tbody>
</table>

Table 5.7 | AOW Architecture. A composition of lightweight technologies
Table 5.7 illustrates all the technologies that AOW uses to design and implement the system. Each of these technologies is lightweight. In addition, there are other low-level design decisions that contribute in this aim like object creation and lightweight structures.

Object creation and allocation. Singleton pattern is used to ensure that certain objects are created only once and reused when needed. Flyweight pattern is used to make sure that heavyweight objects (like REST Client) are created only when all the existing ones are in use. Factory pattern is used to separate object creation from implementation when the creation of the object is complex (in case of Flow, Action and Framework Command Objects)

The low-level design enables the usage of lightweight structures that make better resource management like linked lists and Hash Maps.

- **It is not restrictive in environment technology choices.** Indeed, this application fully supports and encourages open source solutions.

It can be integrated with different technologies. For example, Talend ESBis suggested in the architecture to support AOW integration, but it is not a system constraint. Firstly, an ESB can be introduced only when needed. It was not used for the simulation of the evaluation scenarios. Secondly, instead of Talend ESB, other solutions might be utilized. The only prerequisite is that they should support REST.

This system is implemented in Java, which is a platform independent language. Therefore, it can be deployed in any application server. In this architecture, it is suggested Apache Tomcat as a lightweight alternative, but other options can be used like WebLogic. In addition, it works well both on Windows and Linux. However, Linux is the suggested environment as the log monitoring is easier.

It can be integrated at the same time with different databases. The only prerequisite is the configuration of the database on the server level.

- **It offers simple capabilities for business process modelling.** The end user should be able to create the workflows by performing some simple configurations or running simple scripts.

Usability testing should be conducted to evaluate this claim properly. In section 5.2.4 it is demonstrated that AOW offers the capabilities for workflow modelling using the Data Model
component. The six evaluation scenarios where modelled using configurations and SQL Scripting. The usability testing should answer the question whether these capabilities are simple or not.

The Usability testing is not conducted as it is not possible at this stage of the project for the following reasons:

- The configurations are performed directly in the database and there is not any visualisation of the workflow provided by the system. The graphical interface is needed.
- Automatic semantic checks, loops, and Control Flow vs. Data Flow checks are missing. So the user has to identify configuration errors while debugging the flow.
- The system capabilities are not documented, and there is not any guideline how to construct a workflow using AOW Data Model.

Therefore, the usability testing is part of the future work.

However, the arguments below can be depicted by observing scenarios configuration:

- For these scenarios the business rules are simple. As a result, they are naturally translated in Next Action Rules and Pick Flow Rules.
- There is a one-to-one mapping between actions and business processes operations. Therefore, this translation is straightforward.
- The Data Model does not impose a general guideline on how to build the workflows. As a consequence, the user has the freedom to design the workflows based on the needs. For example, the five evaluation scenarios in Customer-Based Segmentation are designed in one workflow. On the other side, the system does not provide restrictions that prevent the user to make configuration errors, which is a limitation.

As a result of this analysis, AOW pilot lays the foundation for simplicity in workflow modelling, but it does not fulfil it on this stage.

Conclusions on Lightweight evaluation: AOW framework is a composition of lightweight technologies. It does not impose restrictions on technology choices for setting up the environment that it will operate in. For the moment, it is not a simple tool to be used, but it is promising regarding simplicity in workflow modelling.
5.6 Evaluation of Reusability

This project aims to achieve two levels of reusability:

> Service and Framework Command reusability (Service Level)
> Action reusability

Service reusability is evident. The framework offers four generic services (Flow, REST, MAIL, and Database). These services are general implementations of operations. Framework Commands are specific implementations of these services. As a consequence, one service is used by many Framework Commands. In general, all the activities of the business processes will reuse these four services. In the next two sections will be analysed the reusability of Framework Commands and Actions using the modelling of evaluation scenarios.

5.6.1 Reusability evaluation of CBS scenarios

In section 5.2.2 there were addressed 5 CBS Scenarios. As already mentioned, all these scenarios were modelled by reusing the workflow defined for the construction scenario. In this case, we have 100% reusability in Actions and Framework Commands. What is more, we have full workflow reusability including Flow and next action Rules.

These scenarios represent common Marketing Requests in Telecommunication. CBS is considered a subdomain in this domain. This is not a common situation for all enterprises. However, these scenarios are selected to demonstrate the ability of the framework to support the modelling of generic workflows by reaching 100% reusability. In this case, telecom benefits greatly from this design. All the new marketing requests in this subdomain are being addressed with the expected velocity, offering so the best time-to-market.

5.6.2 Reusability evaluation of Loyalty Scenario

The Loyalty scenario is addressed with the modelling of three workflows detailed in section 5.2.3. Table 5.5 and Table 5.6 are used to calculate the reusability of Actions and Framework Commands. AOW maps each workflow operation with one action. In this scenario, each action is assigned with one Framework Command. Therefore, each operation is mapped with one Framework Command. In total (for all the three workflows) there are 15 operations.

From the Framework Commands listed in Table 5.5 and Table 5.6 only 3 are new ones (Add Loyalty Points, Check Bonus Threshold, Send Email). All the other Framework Commands have been already defined to model CBS Scenarios. Instead of modelling 15 new Framework
Commands for 15 operations, there are created only 3 new Framework Commands. Therefore, the *reusability of Framework Commands is 80% (12/15).*

From the Actions listed in Table 5.5 and Table 5.6 only 5 are new ones (Add Points, Get Birth Date MSISDNs, Get Customers Data, Check Bonus Threshold, Send Email). All the other Actions have been already defined to model CBS Scenarios. Instead of modelling 15 new Actions for 15 operations, there are created only 5 new Actions. Therefore, the *reusability of Actions is 67% (10/15).*

From the analysis of Table 5.5 and Table 5.6 and above reusability figures, the following observations are identified:

- The reusability of actions is lower than the reusability of the framework commands. This is an expected result as different actions might execute the same framework command. If the mappings were one-to-one, then the reusability would have been equal. For example Get Subscriber Data and Get Customers Data run the same Framework Command (Get Customer Profile). *(Framework Command Reusability > = Action Reusability)*

- Out of three new Framework Commands, two implement the Database Service. On the other hand, from twelve Framework Commands that are reused, only one implements the Database Service (Add Loyalty Points). Therefore, the Framework Commands that implement the Database Service are not likely to be reused. This is another expected result. The Database Service executes a stored procedure to address an operation specific to the business process that is modelled. *These commands are system-specific.*

- From twelve Framework Commands that are reused, eleven of them implement the REST service. Again this is expected, as they execute services that are provided by the domain. *These commands are domain-specific.*

### 5.6.3 Conclusions on Reusability Evaluation

From the analysis of the 6 evaluation scenarios reusability are identified:

- AOW provides reusability in Service (Services and Framework Command) and Action Levels.
- Reusability in Service level is higher than the reusability in Action level *(Service Reusability > Framework Command Reusability > = Action Reusability)*
100% reusability is possible if generic workflows are modelled for subdomains.

Reusability is increased by encouraging domain specific Framework Commands and avoiding system specific ones.

However, there are some prerequisites to benefit from reusability:

- The ability of the user to identify patterns in a subdomain and to model generic workflows.
- The knowledge of the user on the Data Model Repository. The user should know what Actions and Framework Commands have been designed to reuse them. This pilot does not offer search capabilities on the Data Model. This will be part of the future work.

These results show the reusability for this category of business processes (Marketing Requests in Telecommunication), proving that the framework is promising regarding reusability. Nonetheless, it needs to be further evaluated. AOW had to be exercised in a real world environment and used independently by different developers to model unseen workflows. It is part of the future work.

5.7 Summary

This project is evaluated against its aim and objectives. The simulation of 6 evaluation scenarios demonstrated the AOW functionality for Business Process workflow modelling and execution for this category of business processes.

The accuracy and correctness of the system are evaluated to an acceptable level by means of Unit Testing and code coverage (83.8%), White Box testing and Black Box testing. To improve further this evaluation, the code coverage should be increased to 100% and white box test cases should cover all the test scenarios.

It is evaluated that AOW is a centralised solution based on its SOA architecture and its functionality to offer a single tool for business process modelling and execution.

AOW framework is a lightweight solution as it is a composition of lightweight technologies; it does not impose restrictions on technology choices; it provides the foundation for simplicity in workflow modelling. The simplicity needs to be evaluated further using usability testing.
AOW provides higher reusability in Service level than Action level. Reusability is increased by encouraging domain-specific commands and avoiding system-specific ones. 100% reusability is possible if generic workflows are modelled for subdomains. These results are true for this category of business processes, showing that AOW is promising regarding reusability.
Chapter 6. Conclusions and Future Work

In this chapter are discussed the conclusions of the project, summarising challenges and results. In addition, there are suggested further developments as part of the future work.

6.1 Conclusion

This project aim is the creation of a centralised, lightweight pilot for BPM in the enterprise by using reusable components. To achieve this, it is designed a pilot tool for BPM that uses ECA state based workflow model for business process modelling, and SOA tools for integration in the enterprise. Background reading in SOA, BPM and Workflow models not only enriched the knowledge in this field, but it also provided the foundations of the pilot architecture. Six evaluation scenarios were simulated in a reflection of real world telecommunication environment to evaluate the project against the aim and objectives.

During the project design and implementation were phased many challenges:

- Generalisation vs. Complexity and Usability: Generalisation of Services design introduces complexity in workflow configuration. To encapsulate this complexity, the graphical interphase is needed.
- Flexibility vs. Complexity. This challenge was phased during the modelling of rule based algorithm (Next Action Rules). Introducing system constraints make the rule definition complex and error prone. It works well with simple rules, but it might be difficult to model complex ones.
- Flexibility vs. Controls and Constraints: AOW does not introduce controls and constraints in workflow modelling. Thus, the user can model the workflows based on the needs. On the other hand, the framework does not provide any automatic checks. The users need to perform these checks manuall by debugging the workflows. Therefore, control and constraints should be introduced to prevent user configuration errors.

AOW is exercised in a reflection of a Telecommunication Environment. The analysis of the 6 evaluation scenarios in this domain conclude in the below results:

- AOW provides the capabilities to be integrated into this enterprise environment and to offer the means for business process modelling and execution of marketing requests. For the moment AOW supports only four services that were enough to address the simulated scenarios. However, additional services might be added in the future. Consequently, the service module is designed to be extensible.
AOW is a centralised solution that operates in an SOA using RESTful web services and Talend ESB. It offers a single tool to model, execute and manage business processes.

AOW has a lightweight architecture and design. It provides the foundation to be a simple tool for workflow modelling. However, in the absence of a graphical interface, workflow modelling controls, and guidelines, the framework is difficult to be used in this phase. Usability testing needs to be conducted in the future to further evaluate simplicity in the context of lightweight.

It is demonstrated that AOW provides reusability in Service and Action Level for the evaluation scenarios. If the user identifies patterns in subdomains, the usability may be increased up to 100%. However, it needs to be further tested by independent users in different, unanticipated situations. AOW is promising regarding reusability.

### 6.2 Future Work

This project achieved to design and to implement a pilot BPM tool for the enterprise. However, many important aspects and capabilities need to be addressed as part of the future work to improve this pilot:

- The implementation of a graphical interface is crucial regarding usability. Currently, the configurations for workflow modelling are directly performed in the repository. This interphase should provide the capabilities:
  - Visualising the workflow using standardised flowcharts. In this way, the user will be able to understand the workflow and identify possible errors.
  - Encapsulating configuration complexity. As already discussed many configurations are introduced to make a general design. The interface should make suggestions and provide configuration lists.
  - Providing the guidelines for workflow modelling is essential for the usability. Suggestions and directives will help the user to easily model business processes.
  - Searching the Data Model is a capability that supports reusability. The user should be able to search the created components (Flows, Action, and Framework Commands) and find a description for each of them.
Workflow execution, debugging, and monitoring is performed using logging, which is not convenient. A graphical interface that will manage workflow implementation and monitoring is needed. It should visualise the debug log.

Currently, the framework does not perform any control to ensure that the workflow is correctly configured. Therefore, it needs to implement controls such as:

- Semantic Controls that check the data flow vs. the control flow.
- Controls of rule definition should check if the algorithms return the correct Flow/Action. In addition, these controls should ensure that Next Action algorithm returns only one action at a time.
- Controls for loops and deadlocks should be performed when a new workflow is configured. The user should be able to run this check manually as well.

Management of workflow versioning is a challenge that should be addressed. The system is going to be changed and upgraded many times, but this should not impact the created workflows.

Improving security and performance. The pilot was not focused on these two requirements. Therefore they need to be further addressed and tested.

As already discussed in Chapter 5 the evaluation should be improved as well. Currently, it is performed by simulating predefined scenarios. Part of the future work is the assessment of the system in a real-world environment, tested by independent users. It includes usability and reusability testing.

Adding all these new capabilities will convert AOW framework pilot into an AOW framework tool for BPM in the enterprise.
8 References


[21] E. Y. Mohammad Hossein Danesh, “Analyzing IT Flexibility to Enable Dynamic Capabilities,” in Advanced Information Systems Engineering Workshops, Toronto, Department of Computer Science, University of Toronto, Toronto, Canada, Faculty of Information, University of Toronto, Toronto, Canada, 2015.


and systems science, Stockholm University, 2012.


[44] C. Sabharwal, Java, Java, Java, Missouri : Missouri University of Science and Technology.


Appendix A: Customer Base Segmentation Promotions workflow modelling

Below Figure illustrates the workflow that models all CBS promotions defined in this project (construction scenario and 5 evaluation scenarios). Only one workflow models 6 different scenarios.

Figure A.0.1| Customer Base Segmentation Promotion Workflow
This workflow is translated in configurations of the Data Model components as per below:

<table>
<thead>
<tr>
<th>Flow</th>
<th>First Action</th>
<th>Start Date</th>
<th>End Date</th>
<th>Flow Parameters</th>
</tr>
</thead>
<tbody>
<tr>
<td>CBS Bonus</td>
<td>Get Segment</td>
<td>01.07.2016</td>
<td>01.07.2017</td>
<td>Msisdn, Bonus, Recharge Amount, Bundles, Buckets, Segment Name, Bonus Type, Exists, SMS Test, Command Result</td>
</tr>
</tbody>
</table>

**Figure A.0.2| Components Model, Flow Configurations**
Appendix B: Implementation Details
In this appendix are provided some implementation details of Workflow Engine, Services, Framework Commands and File Reader.

B.1 Database Connection Pooling, Data Source Configuration

Resource specifications in context.xml file:

```xml
<!-- Specify a JDBC datasource -->
<Resource name="jdbc/AOWdb"
  auth="Container"
  accessToUnderlyingConnectionAllowed = "true"
  type="javax.sql.DataSource"
  username="AOW"
  password="7204c97997e631a9c4e1171bd8d7781f"
  driverClassName="oracle.jdbc.driver.OracleDriver"
  url="jdbc:oracle:thin:@localhost:1521:BleOracle"
  maxActive="50" maxIdle="4" />
```

Resource specifications in web.xml file:

```xml
<resource-ref>
  <description>DB Connection</description>
  <res-ref-name>jdbc/AOWdb</res-ref-name>
  <res-type>javax.sql.DataSource</res-type>
  <res-auth>Container</res-auth>
</resource-ref>
```

Data Source Creation:

```java
public void createDataSource(ConnectionConfigurationInterface configuration) throws SQLException, NamingException {
    String dsResourceName = configuration.getResourceName();
    DataSource existDataSource = this.dataSources.get(dsResourceName);
    if (existDataSource == null || configuration.isForced_restart()) {
        Context initContext = new InitialContext();
        Context envContext = (Context)initContext.lookup("java:/comp/env");
        DataSource dataSource = (DataSource)envContext.lookup(configuration.getResourceName());
        this.addDataSource(dataSource, dsResourceName);
    } else {
        throw new SQLException("DataSource \"" + configuration.getResourceName() + \"\") exists and Force Restart=false");
    }
}
```
B.2 Service and Framework Command Execution source code

```java
public FWCommand executeCommand() throws Exception {
    ServiceObjectFactory SOFInstance = ServiceObjectFactory.getInstance();

    ServiceObject serviceObject = SOFInstance.getServiceObject(this.command.getServiceSpecificId(),
                                                                   this.command.getSystemID());

    FWCommand response = serviceObject.executeCommand(command);
    return response;
}
```

`getServiceObject` method:

```java
public static ServiceObject getServiceObject(String ServiceObjectName, String SystemName) {
    try {
        String serviceClassName = ServiceObjectName;

        Class serviceObjectClass = Class.forName(serviceClassName);
        String systemName = SystemName;

        Class[] parameterType = new Class[] {serviceObjectClass, systemName.getClass()};
        Constructor cons = serviceObjectClass.getConstructor(parameterType);
        Object[] args = new Object[] {serviceObjectClass, systemName};
        ServiceObject serviceObject = (ServiceObject) cons.newInstance(args);
        return serviceObject;
    } catch (Exception ex) {
        ApplicationLog.getInstance().writeLog(MainActionTool.SYSTEM_LOG, "ActionSession.getServiceObject",
                                              new Object[] {"ERROR on creating service object", ServiceObjectName, SystemName, ex.getMessage()});
        return null;
    }
}
```

B.3 Example of Triggers translated to a Framework Command

Example of RESTful request:

RESTful Request:

```
XXXXXXXX&BONUS_NAME=MonthlyBonus
```

It is translated to this Framework Command:
AOW | FLOW | RegisterSubscriberBonus | BD3A9F00330C4ADFB495ECE7CD95B846 | Attributes [BONUS_NAME: MonthlyBonus, Msisdn=447XXXXXXXXX]

Response:

AOW | FLOW | RegisterSubscriberBonus | BD3A9F00330C4ADFB495ECE7CD95B847 | Attributes [CommandResult: Success, ResultMessage: FLOWS_ARE_SELECTED, SELECTED_FLOWS: SubscribeBonus, EXECUTE_TIME: 1471788912659]

Example of Pulse:

Pulse Request:

Pulse triggered on: 20160821145001.502

Translated to this Framework Command:

AOW | FLOW | Pulse | 3F2EBF9DDF954B518A6AD6E3134E44D4 | Attributes [TIME: 20160821145001.500 Sun, TIME_MILIS: 1471787401500]

Response:

AOW | FLOW | Pulse | 3F2EBF9DDF954B518A6AD6E3134E44D4 | Attributes [CommandResult: Success, ResultMessage: NO_FLOWS_ARE_SELECTED]

B.3 Example of a RESTful Framework Command:

Request:

AOW|REST| RemoveBucket|Attributes|
RestMethod: get,
RestURI: http://localhost:8080/Online_Charging_System/ocs/RemoveBucket,
ProducedMediaType: application/xml
@queryMsisdn: 447XXXXXXXXX
@queryBucketName: 502

The REST Service Object translates this command into below RESTful Request:
Response:

Rest Service Object returns the below response in XML format:

```xml
<SubscribeBucket>
  <updateResult>Success</updateResult>
</SubscribeBucket>
```

This response is translated into a Framework Command as per below:

```plaintext
AOW[REST] RemoveBucket|Attributes[
  Command Result: Success,
  ResultMessage: 200,
  updateResult: Success]
```

B.4 Framework Command Response: Result Message list

Result Message List:

- RESULT_OK
- RESULT_PARM_ERROR
- RECORD_DOES_NOT_EXIST
- CURSOR_DOES_NOT_EXIST
- RECORD_NOT_CREATED
- RECORD_NOT_UPDATED
- DATE_NOT_ALLOWED
- MSISDN_NOT_VALID
- COUNTER_EXCEEDED
- RECORD_NOT_DELETED
- TOOMANYROWS
- BADCOMMAND
- FLOW_DOES_NOT_EXIST
- END_OF_FLOW
- END_OF_MESSAGE
- LINE_NOT_VALID
- FLOW_NOT_ALLOWED
- RESULT_UNKNOWN_ERROR
- INVALID_BULK_GROUP

B.5 Web Server Request and Response example

Request:

Response:

<fwcommand>
  <commandAttributes>
    <entry>
      <key>EXECUTE_TIME</key>
    </entry>
    <entry>
      <key>SELECTED_FLOWS</key>
    </entry>
    <entry>
      <key>ResultMessage</key>
    </entry>
    <entry>
      <key>CommandResult</key>
    </entry>
  </commandAttributes>
  <commandName>RegisterSubscriberBonus</commandName>
  <serviceSpecificId>FLOW</serviceSpecificId>
</fwcommand>
B.6 File reader configurations

Figure B.1 | File Structure for simulated scenarios

Figure B.2 | File Example

Figure B.3 | File Filers for the construction scenarios