

**FACULDADE DE ENGENHARIA DA UNIVERSIDADE DO PORTO**



**FEUP**

# **Managing System to supervise professional multimedia equipment**

**Luís Soares de Azevedo**

Master in Informatics and Computing Engineering

Supervisor: Professor Miguel Monteiro

18<sup>th</sup> July 2012



© Luís Azevedo, 2012

# **Managing System to supervise professional multimedia equipment**

**Luís Azevedo**

Master in Informatics and Computing Engineering

Approved in oral examination by the committee:

Chair: Maria Eduarda Silva Mendes Rodrigues (PhD)

External Examiner: Daniel Castro Silva (PhD)

Supervisor: Miguel Monteiro (PhD)

---



# Abstract

This document describes the development process of a “Managing System to supervise professional media equipment” and was suggested by MOG-Technologies (Media Objects and Gadgets). The project consists in the implementation of a centralized system capable of providing the user with a unique entry point to control, configure and manage all the different services exposed by MOG’s products.

MOG’s commercial solutions offer features related to video capture, media *ingest* to editable formats and media *outgest* to broadcast extensions. These products are characterized as independent solutions and only allow individual access. This means that in multiple equipment environments, the user needs an extra effort to maintain and take advantage of all the products’ features. Thus, the objective is to present a solution capable of improving this scenario.

The centralized system communicates with all the different equipment by means of Web Services (SOAP) and exposes its features through a Web application. The implemented solution detects and adds resources present in the network, monitors them, provides access to all the media assets reachable by each product and allows multiple configurations and task assignment.

In order to achieve complex workflows between products, a business process definition approach was taken and a Business Process Management System called jBPM was used to design and execute workflows. With this architecture it’s possible to design a sequence of activities using a standard called BPMN2.0 (Business Process Model Notation), which assures interoperability.

With the implemented solution the user doesn’t need to know specifically which features each product has. The only requirement is designing the intended result. It might be capture followed file transcoding to an editable format, the user can simply define the business process with the desired activities and the centralized solution will take care of assigning and scaling tasks.



# Resumo

O presente documento descreve o projeto de desenvolvimento de um “Sistema de Gestão e Supervisão para equipamento de multimédia profissional” realizado na MOG-Technologies (*Media Objects and Gadgets*). A referida solução, consiste na implementação de um sistema centralizado capaz de oferecer ao utilizador um único ponto de acesso para controlar, configurar e gerir os diferentes serviços oferecidos pelos produtos da MOG.

As soluções comerciais da MOG oferecem funcionalidades de captura de vídeo, de *ingest* dos clips para formatos editáveis e *outgest* de modo a ser possível o seu *broadcast*. Estes produtos são atualmente vistos como soluções independentes e permitem apenas acesso individual. Ora, isto significa que em ambientes com múltiplos destes sistemas o utilizador necessita de um esforço acrescido para manter e aproveitar corretamente todas as funcionalidades existentes. O objetivo deste documento é apresentar uma solução possível de modo a melhorar o referido cenário.

O sistema centralizado comunica com todos os diferentes equipamentos utilizando *Web Services* (SOAP) e expõe as suas funcionalidades através de uma aplicação Web. A solução permite detetar e adicionar produtos existentes na rede, monitoriza-los, explorar quais os *clips* disponíveis, configurar múltiplos equipamentos simultaneamente e atribuir tarefas.

De modo a possibilitar complexos *workflows* de interação entre produtos, foi feita uma abordagem de *business process definition*. Para tal foi usado um *Business Process Management System*, chamado jBPM, que permite *design* e execução de *workflows*. Com esta arquitetura é possível definir a sequência de atividades a ser executada pelos diferentes produtos utilizado um *standard* denominado BPMN2.0, o que assegura interoperabilidade.

Com a solução implementada o utilizador não tem de saber quais as funcionalidades de cada produto, apenas tem de descrever o que pretende realizar. Seja uma captura seguida da submissão de um vídeo de baixa resolução num *site*, o utilizador apenas necessita de indicar as atividades na definição do processo e o sistema fica responsável por escalonar e atribuir as tarefas.





# Acknowledgements

I'd like to thank Eng. Miguel Sampaio and Eng. Pedro Ferreira from MOG technologies for the great opportunity and support provided. Also Eng. Rui Amor, Eng. Ricardo Serra and all MOG's collaborators for the time they spent answering my questions, which weren't few or easy. Useful and "on the moment" help is rare and sometimes unappreciated, as such I can't stress enough the support I had during the period I was at MOG. Everyone was willing to offer their valuable time answering any doubts I had.

I also would like to thank the project supervisor, Professor Miguel Monteiro, for all the help and availability showed during the entire process.

Luís Azevedo



# Contents

<b>1. Introduction</b> .....	1
1.1 Context.....	2
1.2 Project.....	2
1.3 Motivation and Goals .....	2
1.4 Thesis Structure.....	3
<b>2. Bibliography Revision</b> .....	4
2.1 Automation Process Brief Evolution .....	4
2.2 Control Protocols.....	4
2.2.1 The 9-Pin Protocol .....	5
2.2.2 The VDC Protocol .....	5
2.2.3 What is RS-422.....	5
2.3 IP Based Protocols.....	5
2.4 SOA (Service Oriented Architecture).....	5
2.5 The FIMS Project .....	6
2.5.1 How does it work? .....	6
2.6 Automated Workflow Systems .....	6
2.6.1 What already exists .....	7
2.7 Media Asset Management (MAM).....	8
2.7.1 Avid Interplay.....	8
2.8 Supporting technologies.....	9
2.8.1 SOAP .....	9
2.8.2 REST.....	9
2.8.3 Message Queuing.....	10
2.8.4 Authentication .....	11
2.8.5 WCF.....	11
2.8.6 Bonjour.....	11
2.8.7 MXF.....	12
2.9 MOG Technologies .....	12
2.9.1 F1000 Software.....	13
2.9.2 S1000 Software.....	14
2.10 Apache Flex .....	15
2.11 Java EE .....	15
2.12 JBoss.....	15

2.13	Ext JS .....	16
2.14	Business Process Management (BPM) Systems .....	16
2.14.1	Business Process Modeling .....	16
2.14.2	What BPMLs are there? .....	16
2.15	Workflow Engines .....	18
2.15.1	jBPM .....	19
2.15.2	Activiti .....	20
2.15.3	Apache ODE .....	22
2.15.4	Intalio BPMS .....	23
2.15.5	Bonita .....	24
2.16	Pros and Cons .....	26
<b>3</b>	<b>Project Specification</b> .....	<b>30</b>
3.1	The broadcast workflow .....	30
3.2	The Problem .....	31
3.3	The Centralized System Workflow .....	32
3.4	Functional Requirements .....	33
3.5	Non-functional Requirements .....	33
3.6	Architecture .....	34
<b>4</b>	<b>Project Implementation</b> .....	<b>36</b>
4.1	Methodology .....	36
4.2	The Implementation process .....	36
4.3	The communication between Web App and Server .....	37
4.4	The Overview Model .....	38
4.5	The Event Hub .....	41
4.6	Control .....	42
4.7	Config All .....	44
4.8	The Settings Module .....	45
4.8.1	Designing a business process .....	46
4.9	The Processes Module .....	50
4.9.1	How processes are run with jBPM .....	51
4.10	Asset Explorer .....	52
4.11	Tests and Results .....	54
4.12	Difficulties .....	57
4.12.1	jBPM Standalone Web Designer .....	57
4.12.2	The Event Hub .....	58
4.12.3	Synchronizing Settings between mxSpeedRail .....	58
4.12.4	“Bubbling” Events .....	58
4.13	Conclusions .....	59
<b>5.</b>	<b>Conclusions and Future Work</b> .....	<b>60</b>

5.1	Goal Achievement .....	60
5.2	Future Work .....	61
<b>References</b>	.....	<b>62</b>

# List of Figures

Figure 1 – Telestream Solution.....	7
Figure 2 – Robots Technology .....	8
Figure 3 – mxfspeedRail diagram .....	12
Figure 4 - F1000 interface .....	13
Figure 5 - Workflow Settings Window .....	14
Figure 6 - S1000 Interface.....	15
Figure 7 - Business processes related standards time-line .....	18
Figure 8 - Web designer .....	19
Figure 9 – jBPM Business Process Designer .....	20
Figure 10 - Activiti tool stack (Liempd) .....	21
Figure 11 - Activiti Business Process Designer.....	21
Figure 12 – Activiti Modeler - Signavio Web Designer for Activiti .....	21
Figure 13 - Business Process in Eclipse BPEL Editor .....	22
Figure 14 - WSDL using Eclipse WSDL plugin .....	23
Figure 15 - Apache ODE web tool .....	23
Figure 16 - Intalio process web app.....	24
Figure 17 – Intalio Designer.....	24
Figure 18 - BonitaSoftware Studio .....	25
Figure 19 - BonitaSoftware User Experience.....	25
Figure 20 - jBPM.....	26
Figure 21 - Activiti .....	26
Figure 22 - Apache ODE.....	27
Figure 23 - Intalio .....	27
Figure 24 – BonitaSoftware.....	28
Figure 25 - Broadcasting Workflow .....	30
Figure 26 - mxfspeedRail interaction flow .....	31
Figure 27 - Centralized System flow .....	32
Figure 28 - Centralized System Technologies.....	35
Figure 29 - Simplified JavaScript ad Web Server communication.....	37
Figure 30 - Overview, Individual Tab with multiple resources.....	39
Figure 31 - Add resource dialog .....	40
Figure 32 - Organize Resources by Group .....	40
Figure 33- Multiple Groups View .....	41
Figure 34 - Global View .....	41
Figure 35 - Event Hub comparison.....	42
Figure 36 - Possible commands to all resources .....	43
Figure 37 - Control group view .....	43
Figure 38 - Synchronizing Settings.....	43
Figure 39 - Config All module .....	44
Figure 40 - Storage Form .....	45
Figure 41 - Locking settings fields .....	45
Figure 42 - Football event workflow .....	46
Figure 43 - Activities .....	47
Figure 44 - Settings Processes Tab.....	47
Figure 45 - jBPM Standalone Web Designer .....	48
Figure 46 - Football coverage event business process .....	49
Figure 47 - BPMN2.0 source.....	49

Figure 48 - Football event workflow preview .....	49
Figure 49 - Processes run .....	50
Figure 50 - jBPM/SOAP interaction .....	51
Figure 51 - Asset Explorer multiple locations .....	52
Figure 52 - Real time asset filter by name and metadata field.....	53
Figure 53 - Adding “New Location 1” to fo2 by drag and drop.....	53
Figure 54 - Test Workflow .....	54
Figure 55 - Missing Assets Warning.....	55
Figure 56 - Asset Explorer and Workflow Interaction.....	55
Figure 57 - Selected Assets (left down corner) for Activity.....	56
Figure 58 - Test Result.....	56
Figure 59 - Workflow Completed Notification .....	57
Figure 60 - Events Architecture.....	59

# List of Tables

Table 1 - BPMS comparison	28
Table 2 - Web Sockets API browser support ()	38



# Abbreviations

AJAX	Asynchronous JavaScript and XML
API	Application Programming Interface
BPMN	Business Process Model Notation
BPMS	Business Process Management System
DBMS	Database Management System
HD	High Definition
HTML	Hypertext Markup Language
HTTP	Hypertext Transfer Protocol
IDE	Integrated Development Environment
JSON	JavaScript Object Notation
MAM	Media Asset Management
MI	Media Infrastructures
MOG	Media Objects & Gadgets
MXF	Material eXchange Format
SDI	Serial Digital Interface
SDK	Software Development Kit
SLA	Service-level Agreement
SOA	Service Oriented Architecture
SOAP	Simple Object Access Protocol
SVG	Scalable Vector Graphics
WSDL	Web Service Description Language
WWW	<i>World Wide Web</i>
XML	Extensible Markup Language
XPDL	XML Process Definition Language
YAWL	Yet Another Workflow Language

# 1.Introduction

The first TV programs were public service oriented with no commercial content and limited air time. In fact, there wasn't any kind of scheduling or other type of automation. Channels were aired for a specific couple of shows, which were manually changed by an operator. The idea of a continuously running schedule with dynamic content and ads based revenue was far away in the future.

This scenario gradually changed with the evolution of electronic components and broadcast methods, which resulted in the increase of TV broadcasters and led to more sophisticated equipment. Not long after, advertisers accepted the TV as a reliable media to publicize their products and the main source of TV broadcasting revenue was born.

Due to the exponential growth of TV content and with the addition of commercials, it became increasingly difficult to efficiently manage the programs' schedules. As such, higher levels of automation were necessary.

"The idea that we can take a manual process and apply technology to it in a way that reduces costs and increases speed, reliability and accuracy is irresistible" (Atherton, 2002). This notion of automation started to have a greater impact on TV broadcasters with the proliferation of computer based systems. By resorting to IT solutions a TV Channel could easily achieve the following advantages:

1. Permanently on the air
2. Improved ads management
3. Data gathering for quality control
4. Reduced staff costs
5. Complex scheduling

These automated systems became very common on TV stations but due to the enormous amount of different tasks involved in this environment, they also got rather complex. With many components and distinct solutions, it was hard to easily interact with all of them. As such, a centralized solution to manage, configure and supervise the equipment is crucial.

From the capturing area to the broadcasting, it's easy to understand that a system capable of automating and unifying the access is an enormous addition. This thesis objective is to define such system.

The subject was suggested by MOG. A company that specializes in creating solutions for TV broadcasting and understands the necessity of increasing its products features and usability.

Regarding this document structure, the remaining chapters are the bibliography revision, where is presented the subject's state of art, followed by the project specification, the project implementation, testing, biggest difficulties and finally the conclusions and future work.

## 1.1 Context

TV broadcasting has been evolving at an astonishing rate with increasing mediums to air from, new data formats, ads management is more complex and viewers want to have access to it wherever they are. Obviously this increases the workload of TV broadcasters' IT department, who needs more technologically evolved solutions. This is where MOG offers its services and products.

MOG was born from a research project at INESC and is presently the world leader in MXF based solutions. Was officially created in 2002 and its core strength is developing TV broadcast systems based on the flexibility of the MXF standard. It's located in Tecmaia(Maia), Portugal, and its main market is international.

Considering the nature of their products, MOG realized that a managing system to supervise their equipment was necessary.

## 1.2 Project

MXF SDK was MOG initial commercial product which intended to provide other organizations access to the referred standard. Afterwards, MOG realized there was a business opportunity in the hardware area.

From this initiative, 3 solutions were developed to date, which are divided into two main objectives:

- Ingest
  - mxfSPEEDRAIL F1000
  - mxfSPEEDRAIL S1000
- Outgest
  - mxfSPEEDRAIL O1000

The issue with this configuration is that interacting with each system must be done individually and considering the amount of activities that require the use of different products, orchestrating the process is a problem. By implementing a managing system to supervise the referred equipment, the performance and usability would be greatly increased.

As such this thesis aim is to design and implement a system able to provide a global view and control of the different products, but at the same time keep the individual interaction depth.

The implementation of the interface was done through the use of HTML5 and JavaScript, while in the backend was used an application server, developed in Java, called JBoss and a business process management system called jBPM. The communication between the Web Server and the resources is accomplished through SOAP and REST interfaces. All of these tools and technologies are explained in the respective sections.

## 1.3 Motivation and Goals

Technologies are getting more and more complex but due to its proliferation user interaction must be kept intuitive and the learning curve shouldn't be steep. If not long ago those who were responsible for editing and managing the TV station content had high IT knowledge, nowadays professionals like journalists want to be able to interact with broadcasting systems easily, even if it is only with the most basic features.

With this paradigm in mind the developed system should provide access to numerous features of a complex distributed system by using the latest technologies in GUI and cross-

platform communication, like HTML5, JavaScript and SOAP, but at the same time all this layers of complexity should be kept away from the user.

This mix of recent and various technologies used and the requirement of keeping things simple, made this project extremely appealing. While it offered a challenge in the technical part it also wasn't trivial to keep the design of such multi-featured system intuitive. One other positive aspect is learning how to approach problems from a business process perspective. With the increase use of business process frameworks, the knowledge acquired during this thesis promises to be very useful.

The main goal of this thesis is to plan, design and implement a prototype of a managing system to supervise and control professional multimedia equipment. The referred prototype should be able to provide monitoring features, by allowing the user to view the different equipment's status, issue direct commands, synchronize settings and plan and execute tasks through the design of workflows. The equipment in question is MOG's mxfspeedRail product line.

During the first part of this thesis a study about the relevant technologies was made in order to understand how they compare to each other and which ones should be used during the implementation phase. A deep comparison between BPMS was necessary as it is the implementation's core technology.

The second part refers to the implementation of a functional prototype and the validation of the thesis objective.

## **1.4 Thesis Structure**

The present chapter offers a contextualization to the thesis subject and its main goals.

The next chapter includes a bibliography revision about the existing technologies and other relevant concepts.

In the third chapter an in depth specification of the project is made including architecture, requirements and mockups.

The fourth chapter focuses on the implementation process, by describing the approach taken, how the system was tested and what were the main difficulties.

Finally, the last chapter presents conclusions and future work.

## **2. Bibliography Revision**

Distributed systems and centralized solutions aren't new and a lot of development has been made in this area. Obviously, the intent of this thesis is not to reinvent the wheel but instead use what already exists about this subject and develop on top of it. Nevertheless, it's relevant to understand how the supporting technologies work, in order to take full advantage of them and choose the right tools.

Another important subject is the evolution of automated system, specifically in the TV broadcasting department. Changing the workflow from almost exclusively manual operations to fully automated environments didn't happen overnight and several technological advances had to happen.

To have a better understanding of these topics, this section provides an overview of the automation process present state, offers a contextualization of the main technologies used on this subject, presents a few commercial solutions similar to the intended result and describes MOG's technologies

### **2.1 Automation Process Brief Evolution**

The early automation systems were a mix of mechanical and electrical system based on a schedule which would determine when to load the next content. These systems were implemented sooner in radio broadcasting and the first ones used in the TV department were rather complex.

"As automation systems improved, the goal was to eliminate the need for a master control operator and let automation initiate events and switch them to air" (VCIS).

With this evolution many new technologies were introduced with some becoming standards used by the major manufactures.

### **2.2 Control Protocols**

As a way to communicate between the participating systems, it was necessary to define an interface to transfer information and rules to determine how the data is packed. To do so, several protocols were implemented and the following two were the most adopted.

### **2.2.1 The 9-Pin Protocol**

The 9-Pin protocol is a two-way communications protocol with various purposes like VCR (Videocassette recorder) and VTR (Videotape recorder) remote control. It's a rather low layer protocol which serves as the foundation for other more specialized protocols, like the VDCP. It was developed by Sony, has a Master Slave methodology and as a communication protocol it has strict requirements regarding data transmission, for instance an Acknowledge package is always sent after the connection is made.

### **2.2.2 The VDC Protocol**

The "VDCP (Video Disk Control Protocol) is a proprietary communications protocol primarily used in broadcast automation to control hard disk video servers for broadcast television" (Harris Corporation, 2011). It's derived from the Sony 9-Pin protocol and uses the RS-422 interface. Although a version of this protocol over TCP/IP interface has been announced the serial cable based implementation is still the de-facto standard. This was due to signal deterioration with long distance communications as the protocol does not include strong data recovery methods.

### **2.2.3 What is RS-422**

The RS-422 is a technical standard which specifies the electrical characteristics of the serial cables used to transfer information. How this information is packed and streamed through the interface is defined by the protocols and both the VDC and 9-Pin are based in the RS-422 standard.

To sum up the RS-422 is the technical standard for the media where the information is transferred using a protocol which can be, among others, the VDC or 9-Pin. Finally the VDC protocol derives from the 9-Pin protocol. It serves as the lowest layer to implement the referred protocols and is the physical support for the communications.

## **2.3 IP Based Protocols**

Although the RS-422 is a widely adopted standard it's not suitable for remote communications over the Internet. These resulted in the TCP/IP protocols.

In order not to dwell too much into these two protocols, because this serves only as an introduction, the following is a brief definition.

TCP/IP or transmission control protocol/internet protocol "describes a protocol which will work on any sort of computer and operating system for transportation of data across the internet between different systems" (Peter, 2003).

## **2.4 SOA (Service Oriented Architecture)**

The basic idea behind SOA is to implement the program functionalities in whatever language suits best and then expose those features as services through a well-defined and platform independent interface, like XML or JSON. This promotes reusability, abstraction and interoperability.

The interoperability is a great feature for the multimedia content management scenario because it would allow seamlessly communication between the array of different components by enforcing an unified interface.

With this architecture it's easy to use the technologies which best serves the purpose and expose the features through web services allowing a language agnostic system.

This is the goal of FIMS (Framework for Interoperable Media Service) which aims to a fully interoperable professional media equipment system.

## 2.5 The FIMS Project

The EBU (European Broadcast Union) and AMWA(Advanced Media Workflow Association) have joined to make the Framework for Interoperable Media Service. Their motivation was the increasing difficulty of interoperability due to different products having distinct interfaces, forcing “the development of custom adapters to integrate components from different vendors” (AMWA Wiki, 2011). Obviously this costs money and time.

In order to remedy these issues, an evaluation of the use of Service Oriented Architecture (SOA) in this environment is being undertaken.

For the manufactures this would mean “reducing their costs and risks associated with integration” (AMWA Wiki, 2011), and for the users means “faster time of integration, with lower cost and risks” (AMWA Wiki, 2011).

### 2.5.1 How does it work?

“The FIMS object model is described by a set of XML schemas, which provide the object model representation for common objects and extensions for the different classes of service.” (AMWA-EBU, 2011). In more detail, being “FIMS Compliant” means that messages between services need to comply with the schemas definitions and “each service interface shall comply with the FIMS WSDL” (Web Service Description Language) (AMWA-EBU, 2011).

As it’s presently defined, the FIMS contemplates three services in its “Abstract Service Layer”, which are Capture, Transform and Transfer. As these are the most common and basic actions within TV content management the framework definition uses them for illustration purposes.

The two main service categories within FIMS are “workflow services able to realize a given business goal” (AMWA-EBU, 2011) and “infrastructure services that are essential components of the Media SOA system” (AMWA-EBU, 2011). In other words, the workflow services define what sequence of actions should be done, for example capture and then transform, and the Media Infrastructure (MI) “conducts the resource allocation, as well as other common services like job scheduling and queuing” (AMWA-EBU, 2011).

Considering that “SOA-based media workflows are often long running process” (AMWA-EBU, 2011), which sometimes can take weeks, it’s essential to have persistence in the SOA BPM (Business Process Model) platform. This feature enables the system to restart at any given point in the workflow without data loss.

In a nutshell, the FIMS has two major layers:

1. The workflow layer which is responsible to orchestrate the services calling sequence, based on the user intentions, and has a persisting state in order to stop and restart the workflow while maintaining data consistency.
2. The Media Infrastructure which contains the Resource Management and is responsible for the scheduling, queuing and allocation of the jobs to be executed.

What FIMS drives to achieve is a way to assure interoperability among different broadcasting products while enforcing good principles on system design based on SOA. Although the framework definition seems to be developing nicely, its success ultimately depends on the manufactures’ will to adopt it.

## 2.6 Automated Workflow Systems

“If the optimized integration of multiple systems in a broadcast process is the objective, workflow is now the solution” (Wadle).

Workflow management was defined by the Workflow Management Coalition (WfMC) as “the automation of a business process, in whole or parts, where documents, information or tasks are passed from one participant to another to be processed, according to a set of procedural rules” (Allen)

These workflow management systems are complex due to the need of accounting for the following aspects when processing the activities sequence:

1. Data persistence
2. Priority
  - a. Some activities might need to be paused in order to more important jobs be processed quicker.
3. Scalability capacity
  - a. The global performance of the workflow must improve with the addition of processing units<sup>1</sup>.
4. Resource allocation
  - a. Load balancing must be assured, meaning that the workload is distributed between the processing units considering performance capabilities.
5. Fall back
  - a. A fall back system must be placed to prevent failure when a not critical component malfunctions.

## 2.6.1 What already exists

### 2.6.1.1 Telestream Vantage®

Telestream launched their main workflow manager called Vantage® back in June 2010. It works as a “server-based software products combine media capture, transcoding, clip management, analysis, QC, and metadata processing into one future-proof video workflow design and automation framework.” (Telestream, 2011)

The workflow is defined using an interface in which each possible action is represented by corresponding icons which are dragged and dropped. First the user sets the “firing” condition, which represents the event that must happen in order to start the workflow, and then using a connection tool, represented by lines, the activity sequence is implemented. A possible result is illustrated in Figure 1.

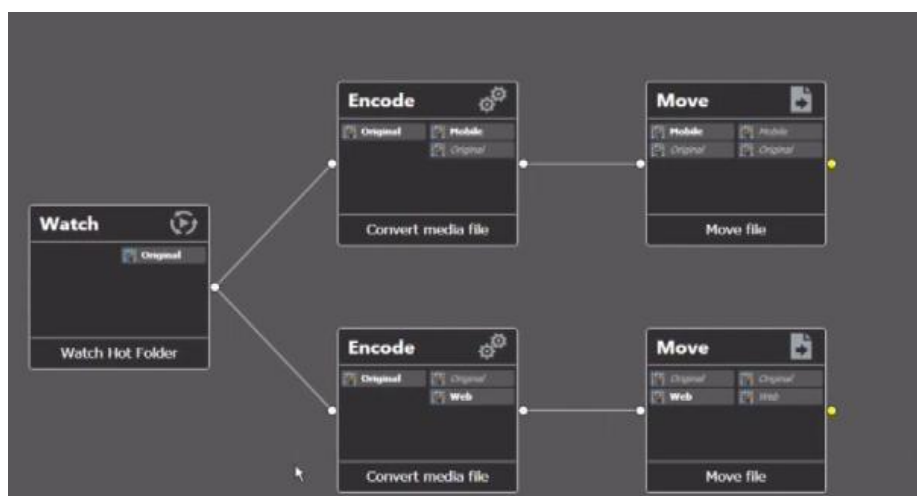


Figure 1 – Telestream Solution

---

<sup>1</sup> In this context processing units refers to the system components responsible for running the present activity



Media files encoding and other aspects of the workflow are defined within the corresponding activities providing an overview of the actions sequence.

Vantage® respects the enumeration previously referred regarding the aspects to be accounted for when implementing the workflow process.

### 2.6.1.2 Robots Technology Content Agent

Robots Technology, which has Telestream as one of their partners, implemented a workflow based solution for media management automation called ContentAgent.

“ContentAgent is a product which provides automated encoding, media management and metadata tracking for many video/audio formats with a complete toolset for automating the transcoding, management and distribution of digital media files. Its user interface has been designed to be operated by non-technical users using a graphical node based workflow tool and it provides a highly efficient encoding engine to create MXF files including frame rate conversion, audio channel mapping, deinterlacing<sup>2</sup> tools etc” (Robots Technology).

A possible state of the workflow interface is shown in Figure 2

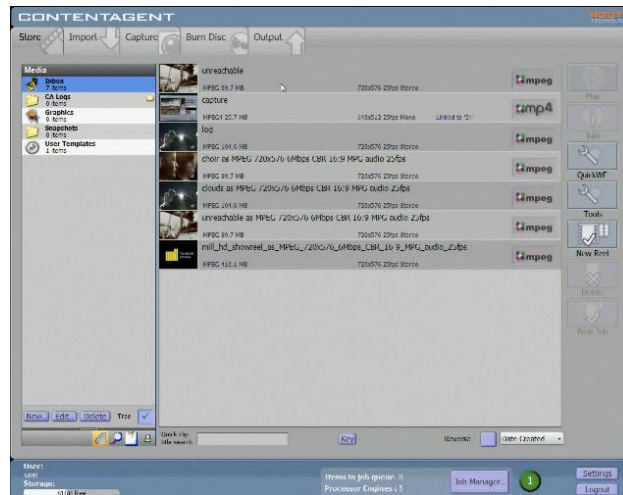


Figure 2 – Robots Technology

## 2.7 Media Asset Management (MAM)

Media Asset Management applications have features to facilitate user interaction with media assets. Operations like cataloguing, storing and finding media files in a large storage environment become more difficult due to increasing data. MAM software tries to resolve those issues by providing options like:

- Indexing assets location so they are easy to find
- Advance search by metadata values
- Categorizing data by type and subject
- Collaborative features

### 2.7.1 Avid Interplay

Avid interplay is a Media Asset Management software which MOG products are compatible with. This decision was made considering the client’s installation base.

<sup>2</sup> “Deinterlacing is the process of converting interlaced video, such as common analog television signals or 1080i format HDTV signals, into a non-interlaced form” (Wikipedia, 2011)

It's currently one of the most used MAM applications with strong features like:

- Traffic system integration
- SOA structure for interoperability
- Web based tools

## 2.8 Supporting technologies

Supporting all these processes in the background, are several technologies which will be analyzed in this section.

Considering that Service Oriented Architecture seems to be a great approach to system designing and it has already been introduced previously. The first technologies to be evaluated support this architecture.

### 2.8.1 SOAP

SOAP stands for Simple Object Access Protocol. The function of this technology is to provide web service evocation and response while using a standardize message format facilitating a Service Oriented Architecture approach.

As previously stated a protocol defines the rules for data changing and SOAP is no different. It's a standard which provides cross-platform compatibility first outlined by the W3C (World Wide Web Consortium) and globally supported by the main software companies.

The message structure used by this protocol is formed by an "envelope" which "wraps" the content, it also has headers and the body message which contains the method to be invoked and the parameters to be used. All of this data is represented using XML (Extensible Markup Language) and usually sent by http (Hypertext Transfer Protocol).

The following is a simple example of a message sent using SOAP:

```
<?xml version="1.0"?>
<soap:Envelope xmlns:soap="http://www.w3.org/2003/05/soap-envelope">
  <soap:Header>
</soap:Header>
  <soap:Body>
    <m:GetStockPrice xmlns:m="http://www.example.org/stock">
      <m:StockName>IBM</m:StockName>
    </m:GetStockPrice>
  </soap:Body>
</soap:Envelope>
```

### 2.8.2 REST

REST means "Representational State Transfer and considers the web as data resource" (Monteiro, 2010). Because of this more specialized focus, it keeps things simpler than SOAP regarding implementation and functionalities.

Usually the services are accessed through URI (Uniform Resource Identifier) which are used "for naming the resources and data items. Operations are performed using the HTTP protocol (and) operation verbs" (Monteiro, 2010). The data is transferred using the format

defined by the service implementation, which might be JSON (JavaScript Object Notation), XML, CSV, or any other format.

Operation verbs are responsible for describing the type of action to be performed on the resources. For example the verb GET serves to “Retrieve the resource identified by the URI” (Monteiro, 2010) as for the POST, it shows intent to “Send (to create) a resource to the server identified by the URI” (Monteiro, 2010). The full list of available verbs is the following:

- GET
- POST
- PUT
  - “Store (to modify) a resource in the server using the supplied URI” (Monteiro, 2010)
- DELETE
  - “Removes the resource identified by the URI” (Monteiro, 2010)
- HEAD
  - “Retrieve metadata (more information) about the resource identified by the URI” (Monteiro, 2010)

In order to better understand how REST works the following URL<sup>3</sup> represents Google’s stock querying service:

[http://finance.google.com/finance/info?q={stock\\_nick}](http://finance.google.com/finance/info?q={stock_nick}) (where stock\_nick represents the stock’s name we want to know more about, being the resource identification)

### 2.8.3 Message Queuing

Most, if not all, Service Oriented systems have the communication between several machines in mind and many of these network elements are spread around the Internet. Although the majority of the internet components, mainly servers, are continually running it might happen that a certain request can’t be fulfilled in a promptly fashion.

For example, a Video Game Retailer has terminals in each shops which register client’s orders and in the event of not having enough stock it sends a request for the Warehouse. Now let’s assume the Warehouse server is powered off during the night time. The orders requests sent by the store can’t simply be “dropped” because the server isn’t available to receive them at the time. Therefore, a way to hold those requests and deliver them to the server at a later time is needed. There’s where Message Queuing comes in.

Message Queuing “provides a mechanism for integrating applications in a loosely coupled, flexible manner by providing asynchronous delivery of data between applications in an indirect way through an intermediary” (IBM).

As there are several approaches to message queue implementation this section will only describe the most common.

#### 2.8.3.1 AMQP

“The Advanced Message Queuing Protocol (AMQP) is an open standard application layer protocol for message-oriented middleware. The defining features of AMQP are message orientation, queuing, routing (including point-to-point and publish-and-subscribe), reliability and security” (AMQP, 2011).

AMQP defines a standard of how the messaging provider and client should be implemented so it’s possible to achieve interoperability respecting SOA principles.

---

<sup>3</sup> URL is a kind of URI

### **2.8.3.2 MSMQ**

MSMQ refers to the Microsoft Message Queuing implementation and it's not an open standard as AMQP. It's obviously "developed by Microsoft and deployed in its Windows Server operating systems since Windows NT 4 and Windows 95. The latest Windows 7 also includes this component. In addition to its mainstream server platform support, MSMQ has been incorporated into Microsoft Embedded platforms since 1999 and the release of Windows CE 3.0" (Microsoft, 2011).

### **2.8.3.3 JMS**

Java Message Service refers to the API used in java applications.

"JMS is an API for enterprise messaging created by Sun Microsystems[...]. JMS is not a messaging system itself; it's an abstraction of the interfaces and classes needed by messaging clients when communicating with messaging systems.

It's used to create a message, load the application data (message payload), assign routing information, and send the message. The same API is used to receive messages produced by other applications." (Mark Richard).

## **2.8.4 Authentication**

When accessing information in a multiuser and distributed environment, security and access privileges are always a concern. LDAP is a possible solution for this scenario.

### **2.8.4.1 LDAP**

If there is the need to implement a system which provides accessibility to a distributed directory information service over a network, then Lightweight Directory Access Protocol is a good option for the application layer.

To, assure compatibility this protocol is defined using Abstract Syntax Notation One which "is a standard and flexible notation that describes rules and structures for representing, encoding, transmitting, and decoding data" (International Telecommunication Union, 2011). The encoding used by ASN.1 is based on BER(Basic Encoding Rules).

Security is the responsibility of the Bind operation which "establishes the authentication state for a connection" by sending "the user's DN(Distinguished Name) and password in plaintext, so the connection should be protected using Transport Layer Security (TLS)" (AMQP, 2011).

## **2.8.5 WCF**

"The move to service-oriented communication has changed software development. Whether done with SOAP or in some other way, applications that interact through services have become the norm. For Windows developers, this change was made possible by Windows Communication Foundation (WCF). First released as part of the .NET Framework 3.0 in 2006, then updated in the .NET Framework 3.5, the most recent version of this technology is included in the .NET Framework 4. For a substantial share of new software built on .NET, WCF is the right foundation." (Chappell, 2010)

## **2.8.6 Bonjour**

"Bonjour, also known as zero-configuration networking, enables automatic discovery of computers, devices, and services on IP networks using industry standard IP protocols. It is a key component of Apple applications (e.g., iTunes, iPhoto), services (e.g., MobileMe) and devices

(e.g., Apple TV, and AirPort). Developers can easily leverage Bonjour from both OS X and iOS” (Apple).

### 2.8.7 MXF

Considering that MXF is the core technology of MOG’s products and the reason why the company exists in the first place, a definition follows:

“The MXF file format enables the carriage of both the audiovisual material and its related information. This ranges from structural information of the material such as compression settings, to geo-localization information, to general descriptive information including transcripts of the content. This why it represents more than a file format. It is a technology that enables the gathering of crucial information as the audiovisual material travels through the workflow. It is therefore the foundation technology driving the adoption of information Technology in the professional media market”. (MOG Solutions, 2006)

## 2.9 MOG Technologies

MOG specializes in the development of systems used to produce and manage multimedia content.

Currently there are three products, which can be grouped by its capabilities of ingest or outgest. Ingest refers to the process of transferring and rewrapping a file. For instance, a file is capture by the S1000 through SDI (Serial Digital Interface) and is sent to file storage. Then, the F1000 is capable of rewrapping the asset so it can be used in the editing room. Both S1000 and F1000 have ingest features. On the other hand there is the O1000, which is responsible for outgest operations. If, for example, a TV station wants to broadcast stored media, it might be necessary to convert the files from the editing enabled wrapper to the transmission format, which is done through outgest and respectively the O1000.

All of this outgest and ingest features are based on the MXF standard which is responsible for the file level wrapper.

To sum up the S1000 is able to capture video from SDI or HD-SDI and save it to a storage server. The F1000 has the capability of wrapping saved files so they can be edited using edition tools. Finally the O1000 makes the file understandable to the play out equipment.

A simple diagram to better illustrate how the system works, is shown in Figure 3.

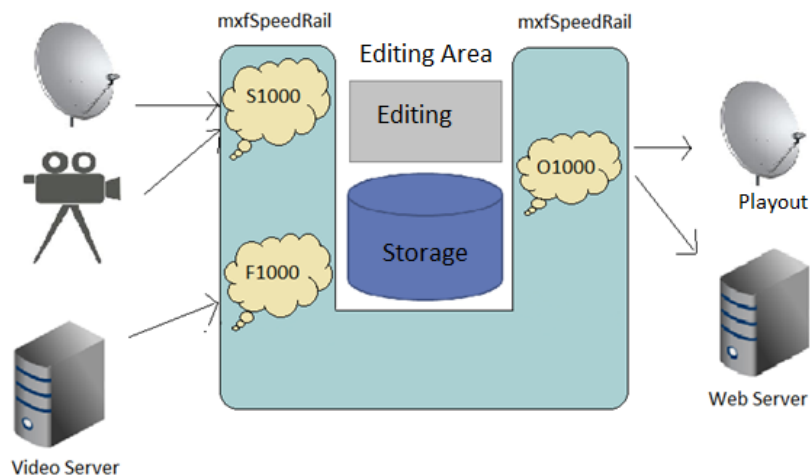


Figure 3 – mxfSpeedRail diagram

While F1000 and O1000 have similar interfaces, as the only differencing aspect between both solutions is the operation applied to the files, S1000 software is quite different. Due to this similarity only F1000 and S1000 interfaces will be shown.

## 2.9.1 F1000 Software

F1000 is a file based solutions and its main feature is *ingesting* stored media into formats compatible with edition software. This allows broadcasters not to worry about compatibility issues between the recording equipment and the software used in the edition environment.

F1000 accepts the following file formats as input:

- MXF OP1A
- QuickTime
- XDCAM

The interface, depicted in Figure 4 , is composed by four main areas: the assets explorer(1), activity controller(2), the workflow profiles(3), Job Control(4) and the notifications area(5).

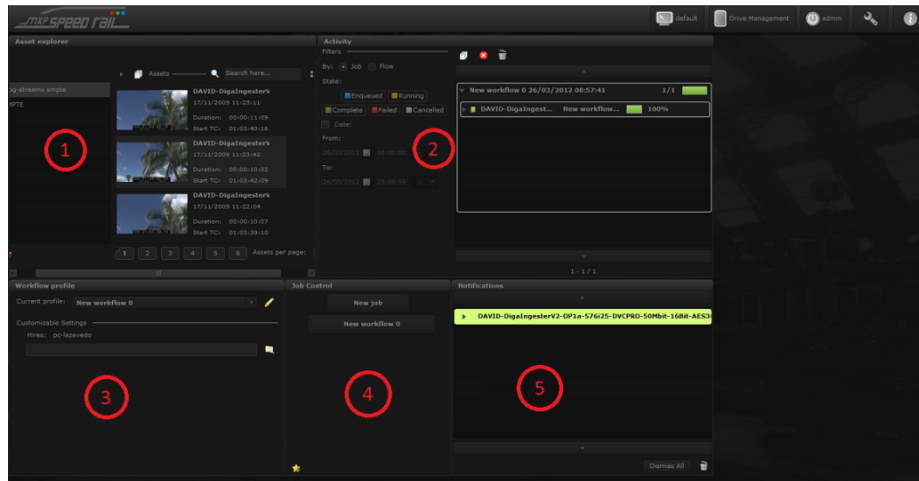
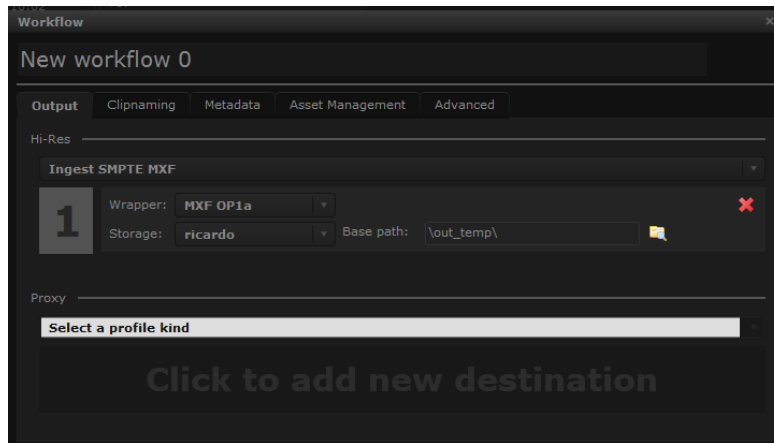


Figure 4 - F1000 interface

1. The assets explorer allows the user to choose files from multiple locations
2. The activity controller displays the current tasks statuses, which can be *enqueued*, informing that the task is waiting to be processed, *running*, *complete*, *failed* and *cancelled*.
3. Workflow profile is where the user defines the ingest operation settings, as seen in Figure 5, which includes:
  - a. The output tab where the user sets the type of the file to *ingest*, the wrapper and the storage which is where the *ingested* media will be saved to. There is also the possibility of generating a proxy version of the standard output, which is basically a file with lower quality, which can be used on the internet for example.
  - b. The clip naming tab is related to how F1000 should name the outputted files
  - c. Metadata tab is where the user chooses the metadata profile, which in turn sets what information should be added to the file, like duration, frame-rate, name, etc.
  - d. Asset Management is related to MAMs, it's where the user sets options associated to Avid Interplay software and other supported media asset managers.

- e. Finally the advanced tab provides options to set the number of sound channels, if the original file should be deleted after *ingest* operation, among others.
4. The Job Control allows the user to start a Job, which is a group of workflows, or a single workflow
5. The Notification Area is a simple history of the result of each Job/Workflow, displaying error or success messages.



**Figure 5 - Workflow Settings Window**

In order to perform an *ingest* operation, the user has to choose one or more assets, select a workflow profile and press one of the Job Control's button. This obviously requires that the workflow profile, storage and location settings be already defined.

To sum up, F1000 fetch a file from a location, applies a workflow profile and saves the result to a storage server.

## 2.9.2 S1000 Software

S1000 is used to capture video from a SDI or HD-SDI signals and *ingest* the media to a storage server. The goal is to assure the media files are compatible with the most common edition software's, this means the supported output formats are:

- QuickTime, used in Final Cut Pro.
- MXF OP1a, for Sony products
- MXF OPAtom for Avid

Figure 6 shows the S1000 interface composed by the monitor area, which displays the media being captures, the profiles area, where the ingest settings are specified and the Clip List refers to the time codes.

One of the major advantages of the S1000 is allowing the file to be edited while the capture is still in progress, saving precious time.

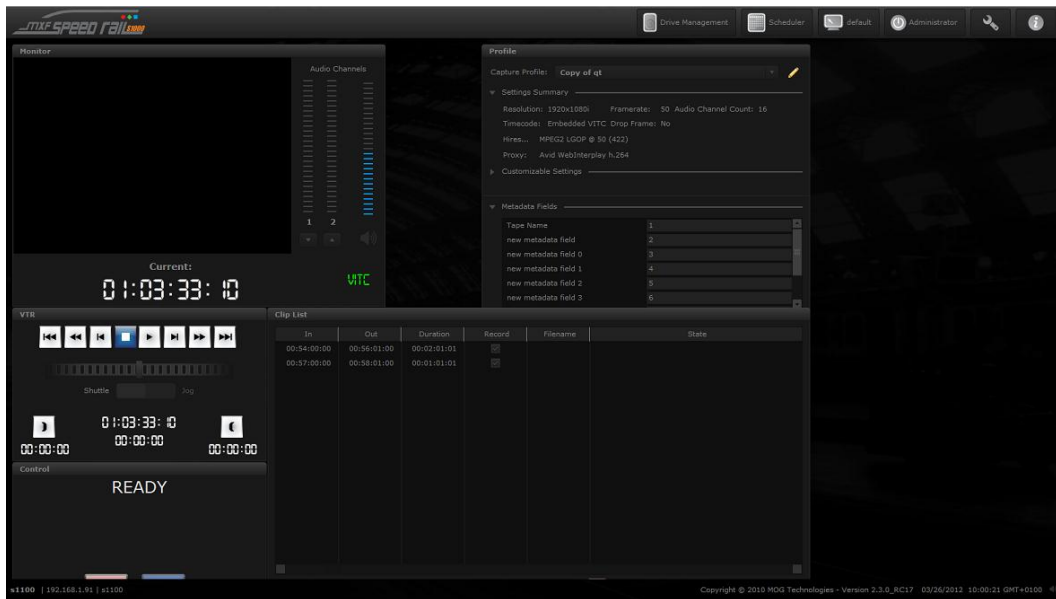


Figure 6 - S1000 Interface

## 2.10 Apache Flex

Previously known as Adobe Flex, Apache Flex is a framework used to implement multi-platform rich internet applications. It is compatible with Browsers because it uses Adobe Flash as the runtime and for desktops it uses Adobe Air 3. Recently it was donated to the Apache Foundation.

Flex SDK provides a set of prebuilt functions and assets that improve development speed, with greater emphasis in interface implementation.

## 2.11 Java EE

What is commonly called Java is in fact the Java SE (Standard Edition) version which contains the basic Java API. Java EE (Enterprise Edition) includes a more extensive set of tools like, Java Servlets, web services, persistence API, among others, and is implemented using a modular approach running on application servers.

Each server application might differ in how the aspects of the Java EE standards are implemented but they all commonly have database abstractions API, web services support and multiple processes optimizations.

Some of the known Java EE implementations are GlassFish Server, Apache Tomcat Server, Apache Geronimo and Jboss. Some of these tools can be found on <http://www.oracle.com/technetwork/java/javaee/overview/index.html>.

## 2.12 JBoss

JBoss is a Web Application Server developed in Java, implements Java EE API, is open source and was recently acquired by a company called Red Hat. JBoss has various frameworks built-in like RestEasy, which is used to make REST request easier and was extensively applied during the implementation.

Web Applications are implemented through the use of Java Servlets, which are responsible for Http requests and responses, JavaServer Pages which allow joining HTML and



Java code and static files like images and JavaScript sources. The Web Application is archived in War (Web Archives) which are executed by JBoss.

## 2.13 Ext JS

Ext JS is a JavaScript framework which can greatly improve the development speed by supplying pre-built models like, Image Viewers, Grids with sorting features, Hierarchical structures, among others. It also has excellent Ajax features to load, delete and update data from servers which for a real time Web Application is very useful.

It has an Object Oriented structure and uses JSON and XML to exchange data with the server. The official web site is [www.sencha.com/products/extjs/](http://www.sencha.com/products/extjs/).

## 2.14 Business Process Management (BPM) Systems

Considering these thesis's goals, the Business Process Management is an excellent approach because "Business Process is a sequence of tasks that happen in a repeatable order, executed by humans and/or systems to achieve a business goal" (Salatino) which in other words is a workflow. This workflow is responsible for translating which activities the different equipment must execute.

### 2.14.1 Business Process Modeling

(Note: From this point on BPM refers to Business Process Modeling and not Business Process Management, except when stated otherwise).

BPM, in systems engineering is the activity of representing processes of an enterprise where processes are "a collection of related, structured activities or tasks that produce a specific service or product (serve a particular goal) for a particular customer or customers. It often can be visualized with a flowchart as a sequence of activities." (Wikipedia, 2011) So a business process model is, in its essence, a workflow.

**But why get the complications derived from business processes modeling to describe a workflow?** Because a positive secondary effect of BPM is using business process modeling languages<sup>4</sup> to describe them, which in turn gives us standards and with that we get the so much coveted interoperability.

BPMLs might standardize the graphical representation of the workflow, the execution semantics or both. But because BPML will mainly serve in this project to transport the definition of the workflow and not its graphical representation, BPMLs which don't have any mapping of the execution counterpart, like EPC (event-driven process chain), will be dismissed.

### 2.14.2 What BPMLs are there?

#### 2.14.2.1 WS-BPEL

WS-BPEL stands for Web Services Business Process Execution Language which serves to implement business process but "export and import functionality by using Web Service interfaces exclusively" (OASIS). Meaning the input and output is done through web services.

Microsoft and IBM each had their own business process language but when the success of other organizations, like BPMI.org and the open movement led by JBoss and Intalio Inc,

---

<sup>4</sup> Although business processing language also refers to a specific meta-language, in this context BPML represents the general group of programming language tools for business processing model.

started to grow they combined efforts and came up with BPEL. From here an obvious advantage of BPEL can be extrapolated, which is having the support of two very influential technologies organizations.

Although it has a considerable support from apache and the referred companies its implementation is older than BPMN2.0

The following is a list of WS-BPEL's pros and cons:

- ✓ External interaction through web services using WSDL<sup>5</sup> and defines a set of those web services.
- ✓ Facilitates the sending and receiving of messages due to the use of WSDL
- ✓ Supports XPath by default due to the xml nature
- ✓ "Supports structures such as 'if-then-else-if-else' and 'while'" (Tony Andrews, et al., 2011)
- ✗ Does not define a graphical representation of the processes
- ✗ A standardize solution for allocating human actions is absent but other approaches do exist

#### 2.14.2.2 BPMN 1.X

BPMN (Business Process Modeling Notation) is seen as a complementation of BPEL wherein its main function is defining the graphical representation of the business process modeling (in our case also known as workflow) and not the execution counterpart. In fact BPMN has an informal execution mapping to BPEL, however, due to "fundamental differences between BPMN and BPEL" it's very difficult "to generate human-readable BPEL code from BPMN models" (OMG, 2011).

To sum up, "BPMN diagrams express the execution flow of the steps to accomplish a certain goal. Important to note that these models are used for people to people communication." (Activiti)

SO BPMN pros and cons are:

- ✗ There are difficulties "Converting BPMN models to executable environments" (OMG, 2011)
- ✗ Does not provide standard xml serialization
- ✓ Standardize support for graphical notation

#### 2.14.2.3 BPMN 2.0

In order to improve the issue of not having a standardize serialization the 2.0 version of BPMN adds the execution model and respective serialized representation using XML. With this addition it's now possible not only to describe the workflow's graphical component but also its execution. The last revision was official launched on January 3<sup>rd</sup> 2011.

#### 2.14.2.4 YAWL

With so many languages there was still space for "Yet Another Workflow Language" (YAWL). "YAWL is sometimes seen as an alternative to BPEL" but the last one has the advantage of being "driven by a standardization committee supported by several IT industry players" while "YAWL has a single implementation at present". The main advantage over BPEL is providing the possibility to allocate human actions with the use of a work list service.

- ✗ One single tool implementation
- ✓ Supports humans actions allocation
- ✓ Supports XPath also

#### 2.14.2.5 XPDL

The XML Process Definition Language basically defines a XML schema (standardize by the Workflow Management Coalition) to interchange business process definition between

---

<sup>5</sup> WSDL – Web Services Description Language

different workflow products. Graphical and semantics parts of the workflow are supported. As such, it tries to join the best of both worlds by serializing the BPMN graphic presentation and providing an execution layer like BPEL. In fact the XPDL site has cloud applications to convert to BPMN and BPEL.

- ✓ Graphical support
- ✓ Execution support
- ✓ Convertible in BPMN and BPEL

### 2.14.2.6 Sum up

Now that the main BPMLs have been analyzed, it's possible to conclude the following:

- Although YAWL is newer than BPEL and has human task support from day one it's not a solid solution due to the low adoption rate.
- BPMN 1.0 is outdated.
- XPDL, BPMN2.0 and BPEL are the three main descriptions languages

Figure 7 has a global timeline of the description languages.

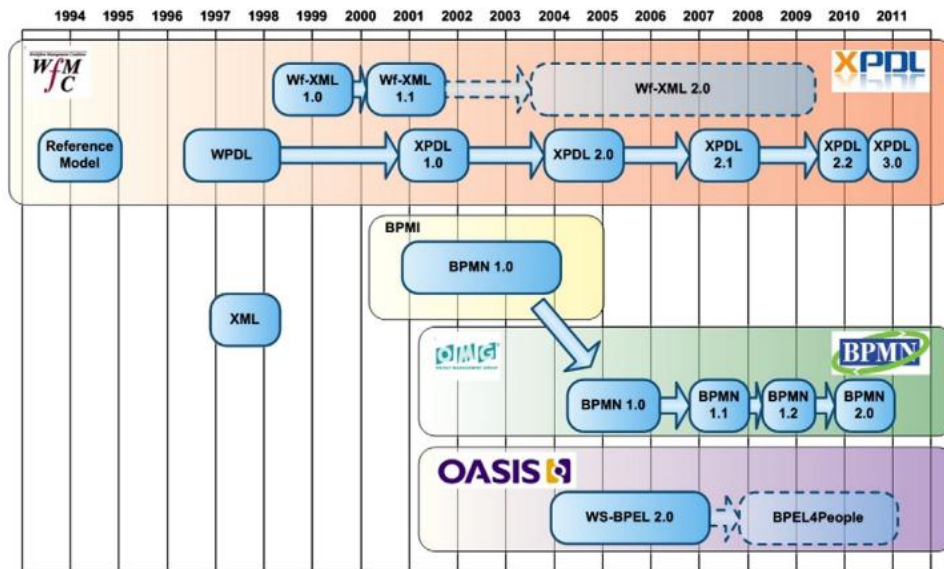


Figure 7 - Business processes related standards time-line (BPMN: An introduction to the standard)

Even if the modeling languages are important the execution tools are more, if the BPPMS doesn't fit the requirements, like external access so it can be embedded in the application, then it won't be used.

The business process modeling language exist to support the BPMS which will be analyzed in the following sections.

## 2.15 Workflow Engines

Obviously the description languages without the workflow engine won't have much use because they serve as means to an end, which is implementing a business process.

“The core of the entire workflow management system is the workflow engine. Workflow engine is responsible to explain the process definition for the implementation of the process instance; scheduling process instance, promoting the process of the work flow.” (Implementation of Process Management and Control Based on JBPM4.4)

With this and the conclusions stated above in mind, a workflow engine evaluation was put together in order to conclude which one is more suited for the task.

## 2.15.1 jBPM

“jBPM is a flexible Business Process Management (BPM) Suite. It makes the bridge between business analysts and developers. Traditional BPM engines have a focus that is limited to non-technical people only. jBPM has a dual focus: it offers process management features in a way that both business users and developers like it” (jBPM) and is written in Java. “The jBPM project has merged with the JBoss Drools project (an open source business rule management framework) and replaced Drools Flow as the rule flow language for the Drools framework” (Liempd).

In a nutshell, jBPM main functionality is modeling a business process using BPMN2.0 and execute it.

In order to define the business process this framework provides four main components:

- Eclipse Editor is a plugin for Eclipse which adds a designer to graphically define the business process
- A web based designer, similar to the eclipse editor but for web browsers. An example can be seen in Figure 8
- jBPM console which allows the business users to inspect and control the process state.
- A REST API to interact with the engine.

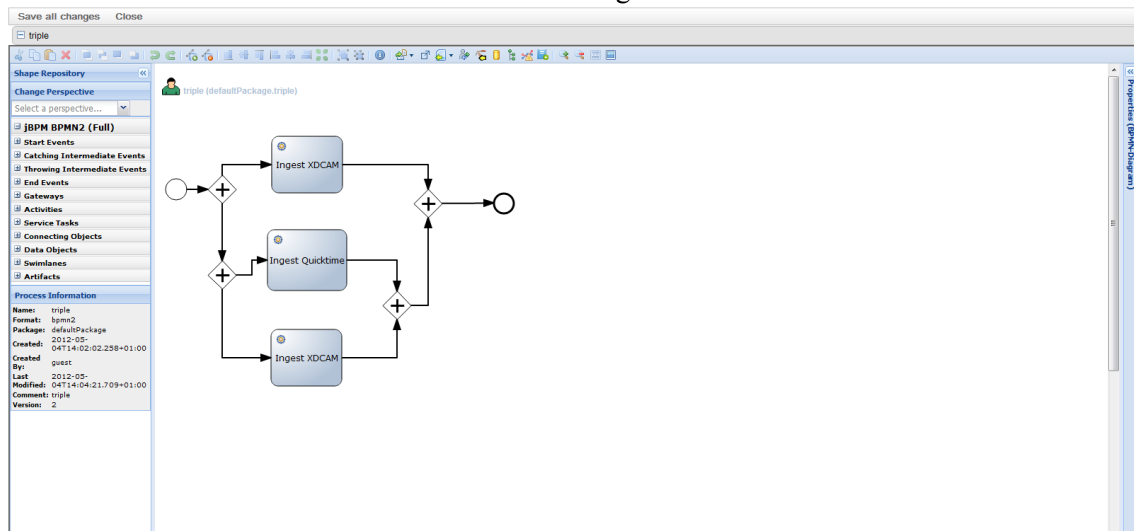


Figure 8 - Web designer

A business process is made of nodes which are connected using sequence flows. BPMN 2.0, which is used by jBPM, defines the following types of nodes:

- Events: They are used to represent the different kind of events included in the business process. Could be a start event or an end event, for example.
- Actions: Are responsible for defining the actions to be performed during the process execution. This is where the actual work of the workflow is represented, for example a human task or sending an automated email to users. Actions can be nested within other actions.
- Gateways: Are used to define multiple paths in the workflow.

In Figure 9 it's possible to see the Events(1), Actions(2) and Gateways(3).

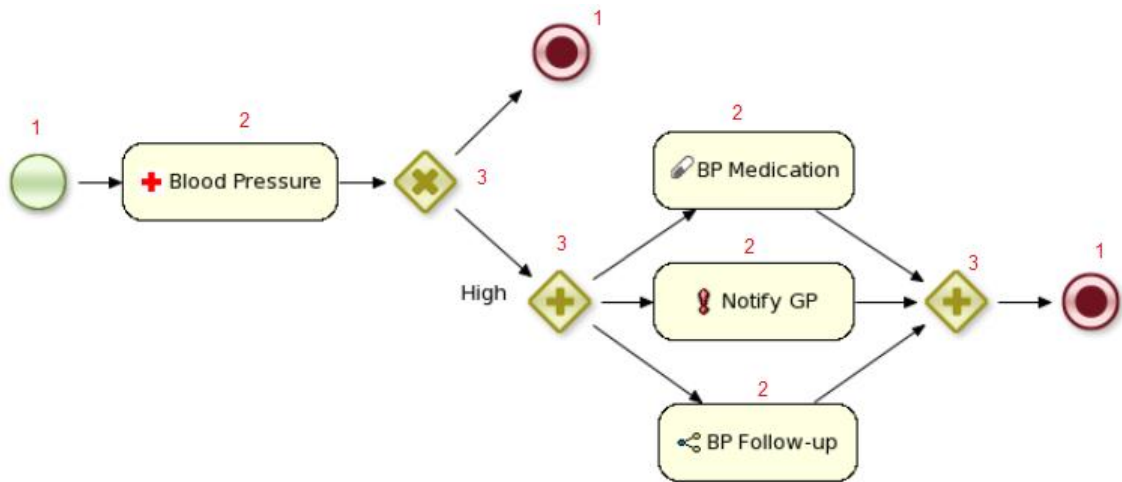


Figure 9 – jBPM Business Process Designer

**How are business process implemented in jBPM?** First a process description must be implemented, either through the designers as in Figure 9, or writing the xml directly. Second jBPM needs to know the process descriptions and for that there is an API component called *KnowledgeBase* which is responsible for maintaining all the process definitions executed by a session. The *Session* is the connection between the process description, included in the *KnowledgeBase*, and the engine. Actions that interact with the process instance, like aborting the workflow, are defined in the session interface. With only these two components is possible to run a business process, but jBPM also has the option of implementing listeners to events, which can occurred at any stage of the workflow.

### 2.15.2 Activiti

“The Activiti project was started in 2010 by the former founder and the core developer of jBPM (JBoss BPM), respectively Tom Baeyens and Joram Barrez. The goal of the Activiti project is crystal clear: built a rock-solid open source BPMN 2.0 process” (Liempd). Due to this fact there many similarities with jBoss but some implementation choices make Activiti different enough to be a serious competitor.

As with jBPM, Activiti also has an Eclipse-plugin to graphically design the business process. It has a web based application called Activiti Explorer which allows users to monitor and control business process but doesn’t provide features to design one. A REST API is also available to support engine related functions from external tools, but is still in experimental phase, meaning “should not be considered stable”(Activiti).

One obvious difference is the web based designer. Although jBPM has a web designer out of the box Activiti relies on an external solution, from Signavio, forcing extra configuration and installation to make it work. This component name is Activiti Modeler.

A summary of the Activiti tool stack is visible on Figure 10.

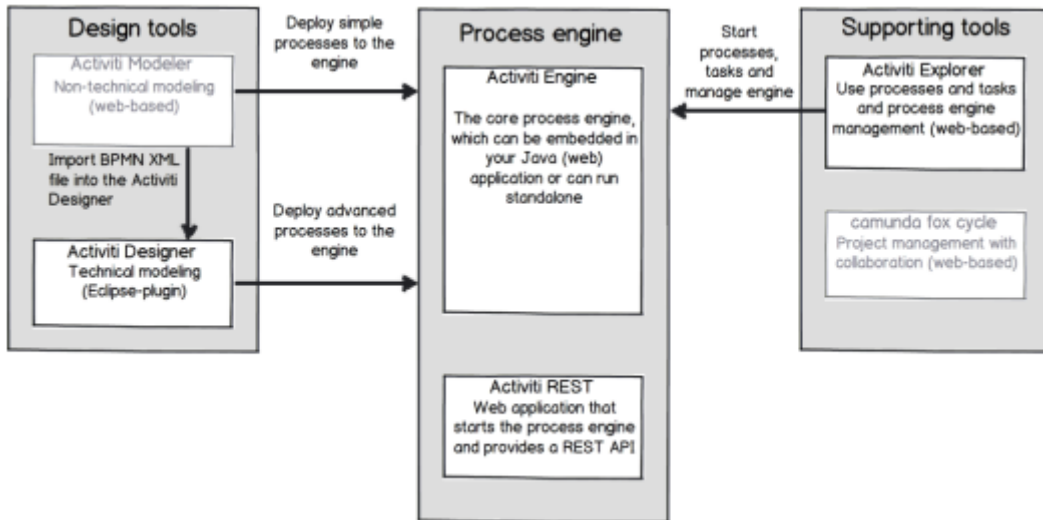


Figure 10 - Activiti tool stack (Liempd)

To compare with jBPM, Figure 11 shows the graphical design plugin for Eclipse. As BPMN defines the visual representation the result is similar to the one in Figure 9, which refers to jBPM.

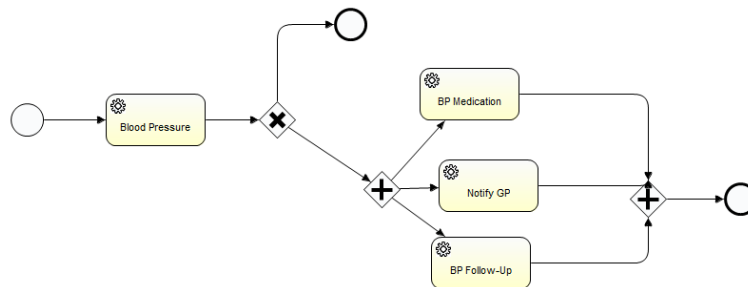


Figure 11 - Activiti Business Process Designer

In Figure 12 is visible a business process modeled using the web designer by Signavio, configured to work with Activiti.

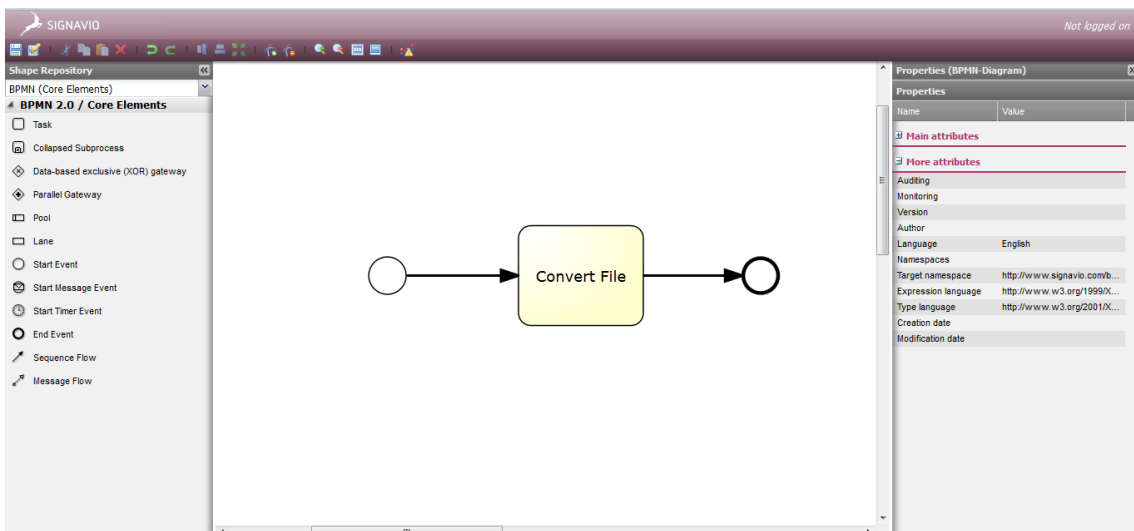


Figure 12 – Activiti Modeler - Signavio Web Designer for Activiti

**How are business processes implemented in Activiti?** As with jBPM, the workflow engine needs a business process description which can be defined using the various components, might be through the Eclipse Designer, writing the bpmn2.0 xml directly or with the Activiti Modeler. Afterwards the model must be deployed using using the Activiti Explorer or Activiti's API.

The Activiti's API has 2 main components, the *ProcessEngine* and the *Services*. The *ProcessEngine* is responsible for running the process business and exposing the *Services*, which are an interface to interact with the Engine. While in jBPM there was a component called *Session* which was responsible for all the interactions with the engine, in Activiti those methods are divided by *Services*. For example if a developer wants to run a business process he must get the *RunTimeService* from the *ProcessEngine*. The same thing happens with the deployment actions, which is made using the *RepositoryService* returned by the *ProcessEngine*. With this structure the methods are more fragmented and not as focused on a single component as in jBPM.

### 2.15.3 Apache ODE

This open source solution from Apache is quite different from the ones analyzed so far. Mainly because, instead of BPMN2.0, Apache ODE supports the WS-BPEL description, meaning there isn't a standardize graphic representation of the business process. As such, the main focus of Apache ODE is the business process execution through the interaction of web services. While Activiti and jBPM have a complex API, the Apache solution relies on the execution description of BPEL, by running the process in conjunction with the data received through web services. This translates in a far simpler solution, installation wised, but the interaction is not as strait forward.

**How are business processes implemented in Apache ODE?** As with the other Workflow Engines the first step is describing the business process, might be using the Eclipse BPEL designer Plugin, like in Figure 13, or writing the xml directly. Then the web services interface must be defined where the interaction and evolution of the business process is exposed. Using the WSDL Editor in Eclipse is an easy way to do it, as in Figure 14. The components defined using the web services description language are the binding between the port where the process service is available, the process itself and the service. The final stage is the deployment descriptor, which is responsible, among other things, to connect the interface with the client.

"The deploy.xml file configures one or several processes to use specific services. For each process, deploy.xml must supply binding information for partner links to concrete WSDL services" (Apache).

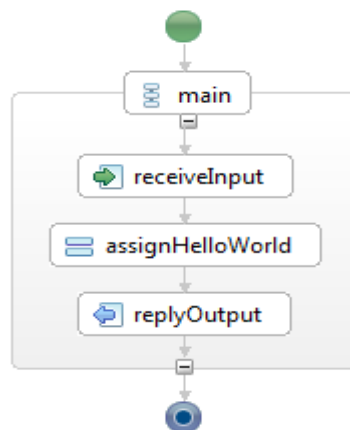


Figure 13 - Business Process in Eclipse BPEL Editor

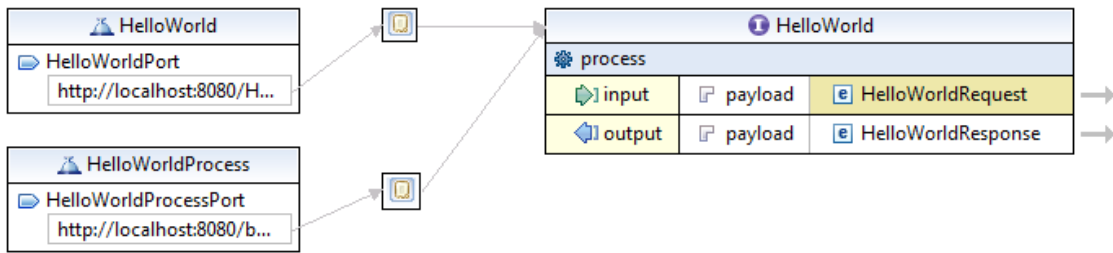


Figure 14 - WSDL using Eclipse WSDL plugin

The web based tool for monitoring the processes is illustrated in Figure 15. It gives an overview of the amount of processes running, displays the .wsdl of each service, has a deployment feature to start new business processes and provides some management hover the running workflow.

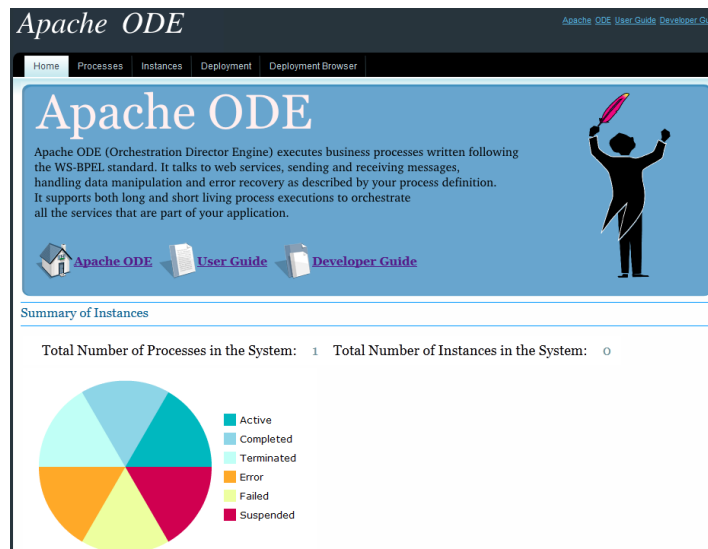


Figure 15 - Apache ODE web tool

## 2.15.4 Intalio|BPMS

“Intalio|BPMS is the world's most widely deployed Business Process Management System (BPMS). Designed around the open source Eclipse BPMN Modeler, Apache ODE BPEL engine, and Tempo WS-Human Task service developed by Intalio, it can support any processes, small or large.” (Intalio).

Intalio is the most commercially driven solution of the ones analyzed so far. In fact, if the free version is chosen only 80% of the source code is available and other features like DBMS compatibility are fewer than in the paid option.

The software is divided in two components, the server and the designer plugin for eclipse. The server package has the actual business process engine which is the Apache ODE. A web app to interact with it, much like the Apache solution, can be seen in Figure 16. A nice detail is the language localization which detects the user idiom, setting it, in this case, to Portuguese.

While jBPM, Activiti and Apache ODE are based only in one language, BPMN 2.0 or BPEL, Intalio uses two. For the graphical design it uses BPMN, but for execution Intalio converts it to BPEL. How this conversion is preformed is not documented and because there isn't any standard for it, figuring how it's made is not trivial. The BPEL code is necessary because Intalio uses the Apache ODE engine, but adds a layer of BPMN with the designer.



INTALIO		PROCESSOS	INSTÂNCIAS	FERRAMENTAS	intalioladmin	ATUALIZAR	LOGOUT	
PROCESSOS								
<input type="button" value="Iniciar"/> <input type="button" value="Ativar"/> <input type="button" value="Reformular"/> <input type="button" value="Implementar"/> <input type="button" value="Não implementar"/>								
Processos	Status	Em Progresso	Falhou	Suspensão	Falhou	Terminado	Completo	Total
<input type="checkbox"/> AbsenceRequest [v1]	Ativo	-	-	-	-	-	-	-
<input type="checkbox"/> AbsenceRequest								
<input type="checkbox"/> HelloWorld [v1]	Ativo	-	-	-	-	-	-	-
<input type="checkbox"/> HelloWorld:HelloWorld								
<input type="checkbox"/> TaskManager [v1]	Ativo	-	-	-	-	-	-	-
<input type="checkbox"/> TMP.TaskManagementProcess	Ativo	-	-	-	-	-	-	-
3 processos	3 Ativos	0	0	0	0	0	0	0
	Reformulado							
<input type="button" value="Iniciar"/> <input type="button" value="Ativar"/> <input type="button" value="Reformular"/> <input type="button" value="Implementar"/> <input type="button" value="Não implementar"/>								

Figure 16 - Intalio process web app

**How are business processes implemented in Intalio?** Intalio relies heavily on the designer when describing the business process. Even the web services layer, as Apache ODE also has, is done graphically. This would be a great way to simplify things but the designer isn't as well developed as the others used in the other suites. For instance while undoing a considerable amount of actions made using the eclipse plugin, some elements appeared even if not added previously, connection arrows and empty tasks, among others. Nevertheless, after having the business process defined using BPMN, the interface layers must be added. For example, if the user wants to monitor the process using the Intalio|BPMS console a new layer must be added to the design. The same for exposing the business process through web services. To better understand how all is put together an example is show in Figure 17, with the interface for the console in the top layer and the web services interactions in the lower one. The binding for how the messages are exchanged is defined using a XML Schema. After all these artifacts are correctly implemented the process is ready for deployment and because the execution engine is Apache ODE this phase is similar to the one described in the Apache solution.

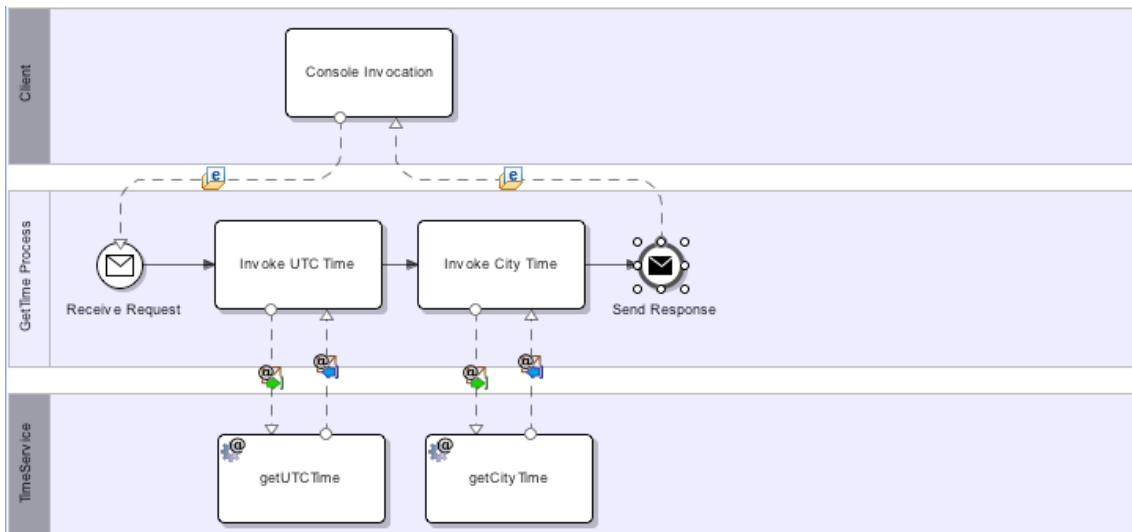


Figure 17 – Intalio Designer

### 2.15.5 Bonita

The most user friendly and intuitive solution analyzed in this section, being clearly more focused on the user than the developer. As such, it's very easy to quickly design and run a

business process but implementing advanced external communication is not very strait forward. Proving is the fact that “BOS [Bonita Software] does not yet provide a public API to build processes so you'll have to develop them using the studio. So no possibility using a web browser yet” (Bouquet).

The studio, shown in Figure 18, is a designer where all the business process implementation takes place. Bonita’s approach with the studio is to define all the process’s aspects through graphical elements, even implementing web services is made using components called *connectors*. By having this structure it’s very easy for a process Engineer to describe and run a business process but at the same time it also limits the possibility of expanding its functionalities, as seen with the constraint of using a customized web designer.

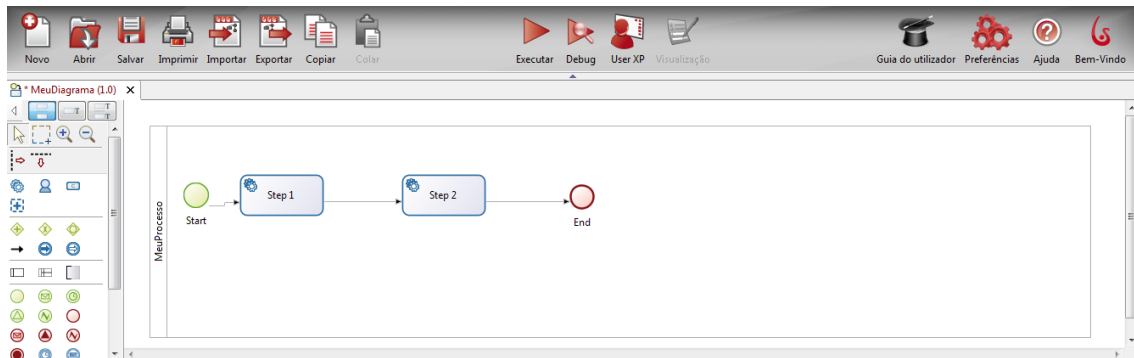


Figure 18 - BonitaSoftware Studio

In order to monitor and control de running business processes BonitaSoftware has the User Experience mode, visible in Figure 19, which is web based and has an email like layout.

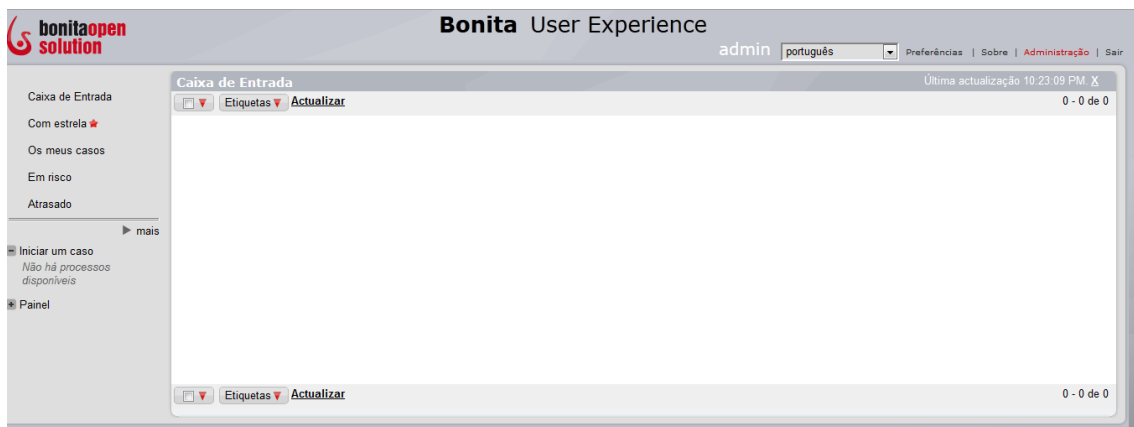


Figure 19 - BonitaSoftware User Experience

**How are business processes implemented in BonitaSoftware?** There really isn’t much to it, the business process can be designed or imported and when is started it will be visible in the User Experience application. All of these actions are made through the self-explanatory GUI. One favoring aspect for BonitaSoftware is being compatible with BPMN 2.0, XPD, or jPDL.

## 2.16 Pros and Cons

In order to have a general idea of the advantages and disadvantages of each BPMS a pros and cons list follows:



Figure 20 - jBPM

- ✓ Maturity
- ✓ Well defined REST
- ✓ Robust eclipse editor
- ✓ Documentation
- ✓ Task Services are straightforward
- ✗ Default web designer isn't very intuitive
- ✗ API is too concentrated, Activiti's service approach is neater
- ✗ Release Cycle not well defined
- ✗ Development team reorganization



Figure 21 - Activiti

- ✓ Should have a better support due to the high number of companies behind it
- ✓ Works well with Signavio Web Designer
- ✓ Well defined Release Cycle
- ✗ Maturity.
- ✗ Experimental REST
- ✗ Documentation not as good as jBPM
- ✗ Activiti uses the Signavio Web Designer and despite being the best web solution it still requires extra configuration.



Figure 22 - Apache ODE

- ✓ It's the "lighter" solution analyzed.
- ✓ The main, and almost only, goal is to execute the WS-BPEL model.
- ✗ Not very straightforward due to an almost none existing API
- ✗ Release Cycle it's not very frequent
- ✗ Doesn't have a standard graphic representation.
- ✗ Poor Documentation and designer support



Figure 23 - Intalio

- ✓ The installation and configuration process is very easy
- ✓ Due to the adoption of BPMN it has a standard graphic representation
- ✓ Execution is done using Apache ODE
- ✗ Community version, which is free, only comes with "80% of open source"
- ✗ The conversion from BPMN to BPEL is unknown.
- ✗ The eclipse designer isn't very robust
- ✗ Web Designer isn't included in the community version



Figure 24 – BonitaSoftware

- ✓ Very easy to install
- ✓ Self-explanatory
- ✓ “Pickup and play” feeling
- ✓ Great GUI
- ✓ Supports XPD, BPMN2.0 and JBPM
- ✗ Doesn’t have web designer
- ✗ Due to the lack of “external” interfaces it isn’t easy to develop an external designer
- ✗ Service Tasks implementation is not easy
- ✗ Relies too much on graphic development, coding customized behavior is not easy

In order to have a better idea on how each BPMS stack together a comparison table is shown on Table 1 where is possible to understand that jBPM is the best option considering the application’s requirements.

	External Interface	Easily Embedded in application	Open source
jBPM	✓	✓	✓
Activiti	✓	✓	✓
Apache ODE	✓	✓	✓
Intalio	✓	✓	✗
Bonitasoft	✗	✗	✗

Table 1 - BPMS comparison



Supported through API



Although possible it requires considerable extra development



Not supported

# 3 .Project Specification

This chapter provides a detailed description of the problem, followed by the solution concept, requirements specifications and system architecture.

It serves as a contextualization to the implementation chapter, restricting itself only to the theoretical concepts.

## 3.1 The broadcast workflow

Conceptually speaking, the fundamental activities needed to broadcast media are acquiring the footage, editing the data and broadcasting the final result. Within these different broadcasting stages there are sub-activities related to file management. For instance, when capturing an event, the data might be saved to a file type not compatible with the editing software. This is an inconvenient related to the number of different capturing solution existing in the market. In order to solve this incompatibility issue, many TV Broadcaster buy MOG products which have features of *ingest* and *outgest*. As explained earlier, *ingest* refers to the steps needed when importing captured data into software, this might include *rewrapping* operations so the final file type is compatible with the application. The *outgest* refers to the inverse process, going from the file based solution to the play out equipment which is commonly based in the serial digital interface.

The ingest capabilities are related to the S1000 and F1000 products. While S1000 supports capture and has some ingest features, the F1000 is *ingest* only but provides a wider range of file types. To play out the final media MOG has the O1000 solution.

To better understand how the different products interact, a common workflow present in the broadcasting environment is shown below:



Figure 25 - Broadcasting Workflow

In the capture activity a S1000 would be used, then a F1000 would *ingest* the stored data to different file types and before broadcasting the result, O1000 would *outgest* the edited data to the play out equipment. All of these products are part of MOG’s mxfSpeedRail line.

This example happens multiple times during the day to day broadcasting process. As such, many TV Broadcasters need several *ingest* and *outgest* solutions from MOG.

On the product level, translating the referred workflow to mxfSpeedRail interactions can be represented by the following diagram:

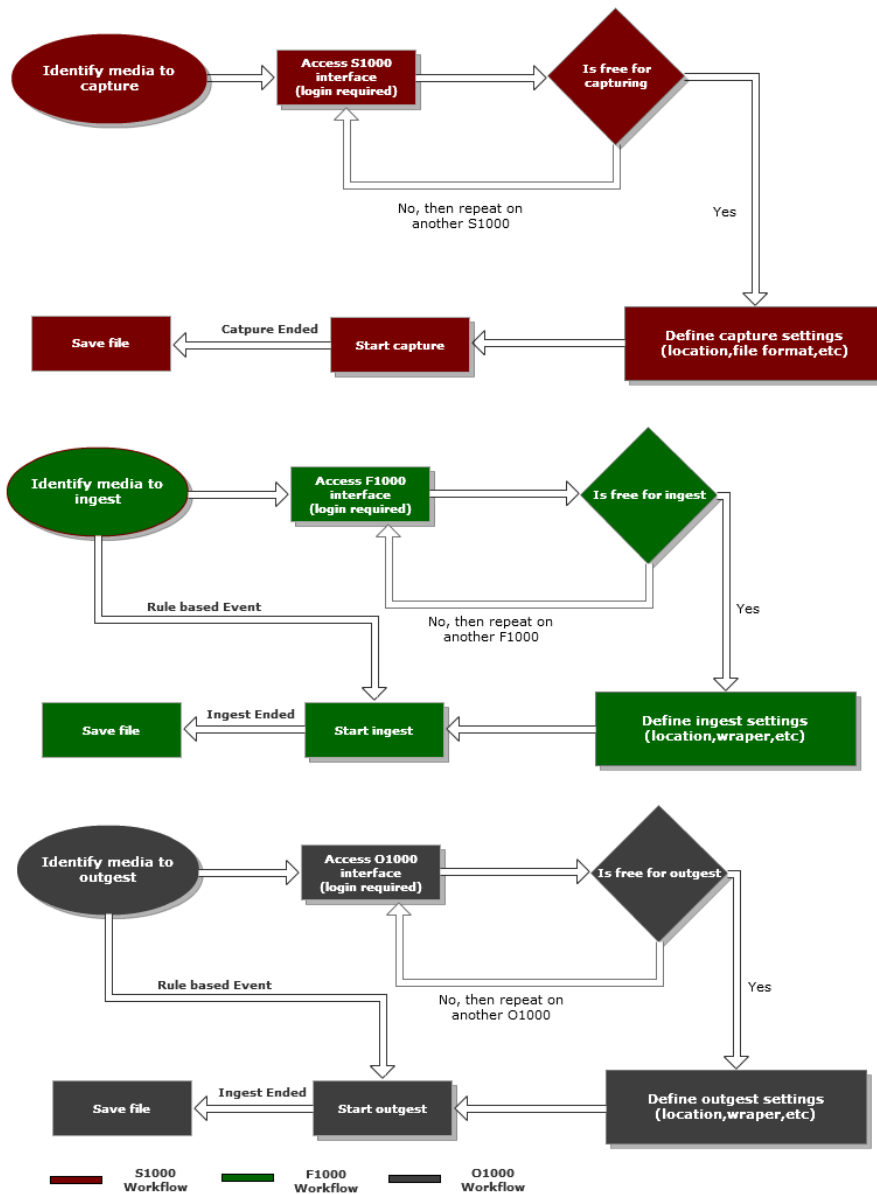


Figure 26 - mxfSpeedRail interaction flow

Basically each workflow can be summarized to input identification, applying a configured operation and saving the file.

Both F1000 and O1000 have *rule based events* which allow the user to specify behaviors on specific events and automate this sequence to a certain degree. For instance, it's possible to configure F1000 in order to automatically start an *ingest* operation when a file is saved to a specific location. This removes the necessity of supervision to assure the workflow proceeds. Nevertheless, these *rules* still need to be set one by one, be it by importing settings or defining them manually.

### 3.2 The Problem

The current mxfSpeedRail interaction flow contains a considerable number of repetitive tasks, which is worsened by the amount of products used by the client.

For example, if a broadcaster has 12 S1000, 20 F1000 and 8 O1000, the above tasks might demand some time to accomplish. The user would have to find among the 12 S1000



which one was free for capturing which might result in 12 authentications, considering the worst case scenario. Besides having this same issue, F1000 and O1000 also have another one related to network discovery. While the S1000 has *Bonjour* running, which improves the machine's visibility in the network, the remaining products haven't. This means that to access one of the twenty F1000, the user needs to know the machine's name or have some tool to browse the network.

Monitoring the execution state of the workflow's activities also gets difficult with so many resources, if there were ingest operations on all of the F1000 it would be hard to keep track of 20 interfaces.

Configuration is also an issue, while the mxSpeedRail products have import and export settings options, this operation has to be done one by one, which makes keeping configurations updated a difficult task.

To sum up these are the main issues with the present system:

- Repetitive tasks
- Difficult to monitor multiple products at the same time
- Knowing what resources are available in the network
- Configuration management complexity increases with resources

### 3.3 The Centralized System Workflow

By adding a new product to supervise and manage the mxSpeedRail product line, the interaction flow can be greatly simplified and optimized. All the user actions would be focused on a single interface which provides access to all the S1000, F1000, O1000 and even future products that might exist in the network.

The Centralized System provides a global view of the mxSpeedRail solutions, where relevant events, like errors, are immediately identified and presented to the user. So, if among 20 F1000 one of the ingest operations fails a warning is shown, removing the need to keep track of each individual state through the various interfaces.

Another issue that the new system is able to resolve is the configuration hassle. Because the user has now access to all the resources, importing and exporting settings becomes a lot easier. For instance, the user could configure a media location in one F1000 and simply replicate the configuration to the remaining products.

To control the actions to be performed by each resource the user must define a workflow with the activities to be executed, for instance a capture followed by an *ingest* and then an *outgest* operation. Load balancing is transparent from the user perspective with the system optimizing how the various tasks are distributed among the resources.

With these features the previously referred three interaction flows can be merged into one:

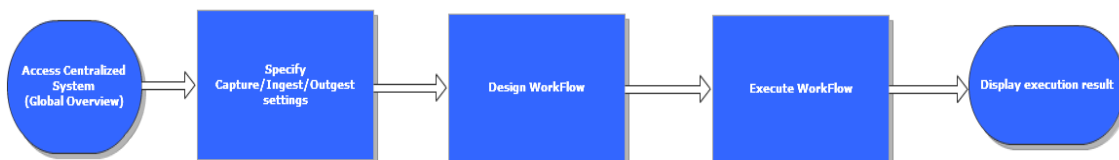


Figure 27 - Centralized System flow

Considering this new workflow the following advantages are easily accomplished:

- Replication is minimal due to the “define once apply to all” feature
- Unified interface with global view of all products
- Complexity is kept low even with increasing number of resources
- Instant information of important events

It's easy to understand that this solution provides a considerable increase in usability and global system performance.

### 3.4 Functional Requirements

With the referred issues in mind and by having a general idea of what improvements the Centralized System needed to offer, a functional requirements specification was done.

The following list presents those functional requirements in descending order of relevance:

1. Add mxfspeedRail products to the Centralized System
2. Display notifications and task history for each resource
3. Organize resources within groups
4. Provide a global view of the added resources, including:
  - 4.1. How many there are
  - 4.2. How many are performing work
  - 4.3. What kind of work is being performed (ingest/outgest/capture)
5. Issue basic actions to all resources, selected groups and individually
  - i.e: A global shutdown command must be possible
6. Synchronize configuration options between resources
  - i.e: Copy an asset location from one F1000 to another
7. Design workflows with tasks to be performed by resources
  - i.e: A workflow with 3 different *ingest* operations to be executed at the same time
8. Execute the workflows

The first requirement refers to the possibility of adding new resources to the Centralized System, be it by specifying the settings manually or by using *Bonjour*.

The organization of resources within groups, referred in the third requirement, was defined to divide the products by broadcast areas. For example, a TV broadcaster has both News and Sports shows, but the first one needs to maintain a defined number of resources ready to capture and *ingest* important events. As such, not everyone should be able to change those resources settings nor should they be assigned with tasks at the time of need. By grouping these resources it's possible to keep them task free and with the settings correctly defined.

### 3.5 Non-functional Requirements

After defining what the system had to do, some requirements regarding performance and usability were set. These requirements cover aspects related to the system behavior and overall experience. For instance, there is no functional requirement stating that the system should be responsive and consistent, but in order to have a well-designed and implemented application that obviously must be accounted for.

The following list covers the non-functional Requirements:

- High level of responsiveness
- Performance
- Compatibility with different Browser versions
- Load balancing

While the first and second requirement might seem the same, they are quite different in the application context. Although the system should perform all the tasks as fast as possible the interface should never hang during execution. For example, while the server side is performing a task the user interface should continue responsive and provide the appropriated feedback after completion.

Because the interface is implemented in HTML5, browser compatibility is a concern and the application should perform well in the following versions:

- Internet Explorer 8 or higher
- Chrome 18 or higher
- Firefox 11 or higher

The last requirement refers to the use of load balancing when assigning tasks to resources. This means that when a new *ingest* task is submitted by the user, the F1000 that will execute it should be the one with less pending tasks.

### 3.6 Architecture

Due to the number of components used in the solution implementation, the system architecture got quite complex. In order to decide which technologies and tools to use, a deep analysis of the system requirements was made.

The interface had to be implemented in HTML5 and needed a web server. Deciding the Web Server type required an analysis of the Business Process Management System in order to assure compatibility. After choosing jBPM, the only possible Web Server was JBoss. This is due to Drools Guvnor requirement of only working with JBoss application server.

To better illustrate the technologies used and how they interact with each other, a deployment diagram is shown in Figure 28.

User interaction is done through the Web Application which will have two models: the Centralized System interface which contains the monitoring, control and share settings features, and a modified version of the jBPM Web Designer which will be used to define the workflow. Although the second model will be integrated in the first one, it is also accessible from outside the Centralized System application as a Standalone component.

The JBoss Web Server will be the main component of the Centralized System. This is where the jBPM API will run the business processes/workflows and interact with the mxfspeedRail stack through SOAP.

Guvnor repository is used to persist and load business process models and related assets. When a user saves the business process within jBPM Designer, the BPMN2.0 xml is directly sent to the Guvnor repository and a SVG representation of the process goes to the Web Server. To retrieve the process list the Web Server communicates through Rest with the repository and if the user wishes to edit the jBPM Standalone Editor is opened with the process definition.

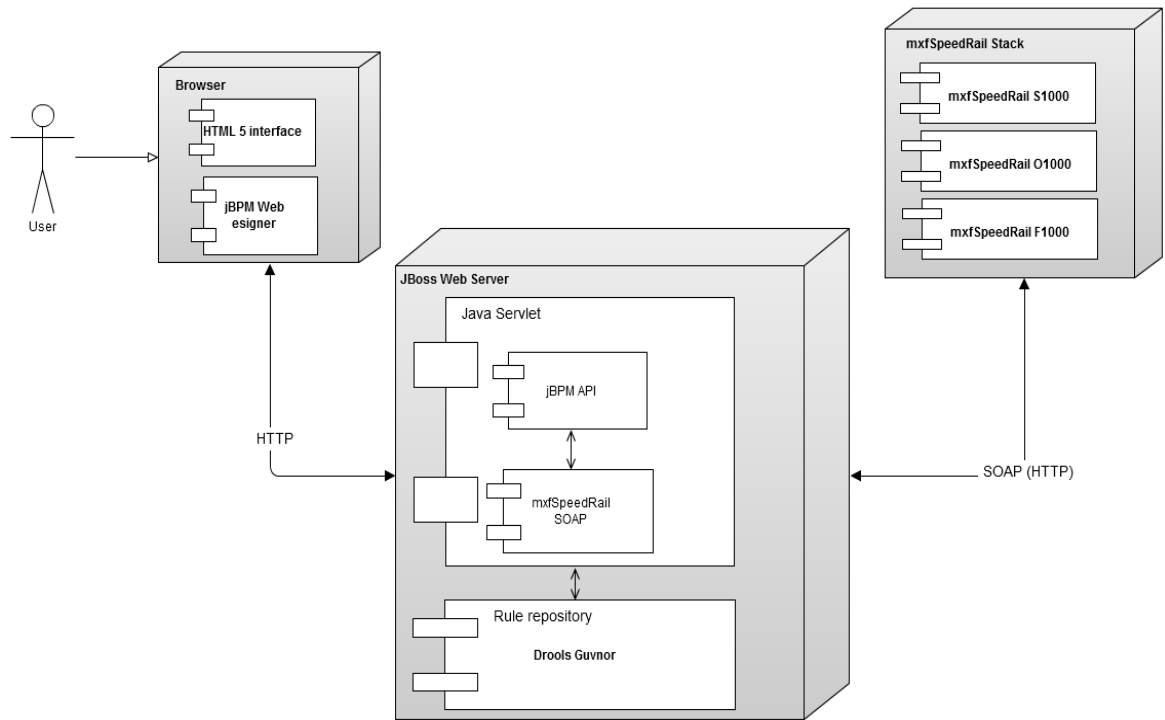


Figure 28 - Centralized System Technologies

# 4 Project Implementation

## 4.1 Methodology

There are several methods to develop software and the challenge is not only knowing them, but also when to use each one. After ruling out several alternatives, only two methodologies seemed suited for this project, namely the Iterative and incremental development and the Spiral model. However there was a favoring aspect towards the iterative implementation which was the easiness of dividing the Application in models and implementing those models through small increments.

The Iterative and Incremental basic idea is to develop the system by implementing small portions at a time through systematic cycles. By doing this the developer is able to better apply the knowledge acquired during previous iterations, concentrate all the attention in the iteration at hand, execute well defined and automated tests on each cycle and do detailed planning.

The development process is divided in the following phases:

1. Initial planning
2. Implementation cycle, which includes:
  - a. Planning
  - b. Requirements
  - c. Analysis and Design
  - d. Implementation
  - e. Testing
  - f. Evaluation
3. Deployment

The first phase is done to assess the general requirements, including non-functional, the system main objectives, scope and risks. Although this phase shouldn't go into implementation details, it should provide enough information to estimate the project duration.

The second phase is the longest and the core of this methodology. The cycle starts by planning the development to be done during the iteration, this is when developers decide where to start, for instance which application model should be implemented. Then the requirements for the iteration are defined followed by an analysis of those same requirements and respective design. This is all done to assure that the implementation phase goes as smooth as possible by relieving programmers from other responsibilities besides coding. After implementation, the application is tested within the iteration scope and an evaluation is done to assess what will carry on to the next iteration.

This methodology also had the advantage of being flexible when requirements change.

## 4.2 The Implementation process

The initial planning was made during the first meeting with Eng. Miguel Sampaio and Eng. Rui Amor, where the core notions and requirements of the prototype were defined. The main conclusion from this meeting was dividing the Web Application by models, which were:

- Overview
- Control
- Asset Explorer
- Config All

- Processes
- Settings

Afterwards the iteration cycle started with the planning of the first model, “Overview”, which led to a preliminary mockup design, followed by analysis and implementation. Due to the systematic approach of the Iteration Process this cycle was repeated through the development process.

To design the mockups during the planning phases, two different tools were used: a plugin for Firefox called “Pencil Project” and a Web App named Gliffy. While the latter was the easiest to use, the former had interaction events which proved to be crucial in the more complex modules. After the mockup was defined, it was validated in conjunction with the requirements and it would be modified or used accordingly.

Before implementing any of the application’s modules, a crucial system layer had to be developed, the SOAP layer. This layer allows the Web Server, implemented in Java, to interact with the mxfspeedRail products, implemented in .Net. To accomplish the task of creating the Java stubs a tool called *Axis2* and another called *Wsimport* were used. These tools parse the WSDL of the corresponding SOAP Web Services to output java classes used during the services evocation. The resulting artifacts were grouped within .jar files and added to the jBoss library as it is depicted in Figure 28.

Eclipse was the chosen IDE due to the amount of different technologies used in this project. Although Eclipse might not be the best for HTML or JavaScript development, it has a balanced performance in all of the programming languages used.

To improve the implementation speed and the application design two JavaScript frameworks were used: JQuery UI, for basic components like dialogs and tabs, and Ext JS4 for more complex interactions and visual components.

Another crucial implementation phase was studying the SOAP services provided by the mxfspeedRail products.

### 4.3 The communication between Web App and Server

Due to the Web Application nature, the user should get a sensation of high responsiveness and performance while using the interface. To achieve a non-statically feeling, Ajax was heavily used on the communication between the Browser and the Web Server. By using Asynchronous JavaScript and XML(Ajax) requests the browser doesn’t need to refresh the page in order to display new content received from the server. To facilitate the complex synchronization operations between the Web Application and Web Server and to increase code reutilization, the following Object Oriented approach was made:

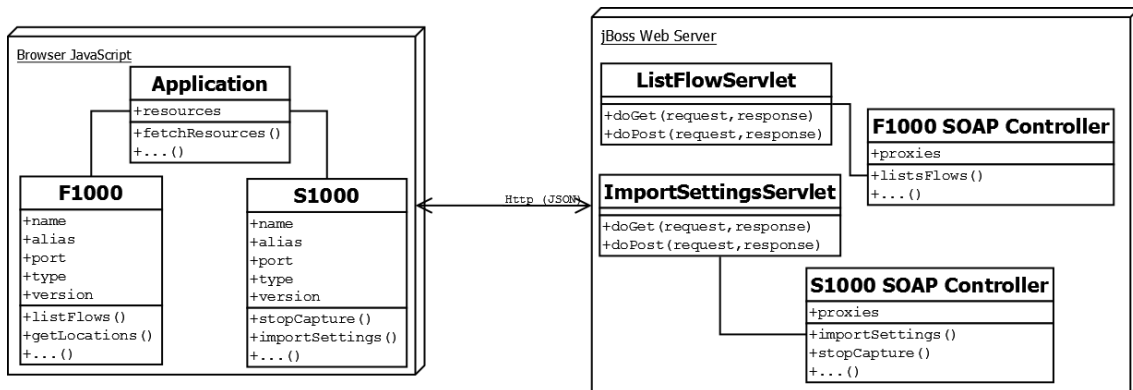


Figure 29 - Simplified JavaScript ad Web Server communication

This simplified diagram shows an Application class in JavaScript, which is responsible for synchronizing the resources between the client and the web server. F1000 and S1000 are the

two resource classes and serve as mirrors to invoke the Web Server methods. For instance, the *getLocations()* method in the JavaScript class is responsible for encoding the request in JSON, invoking the mirrored server *getLocations()* method through Ajax and returning the result to the caller function through a *Callback* system. By using this architecture a Web Application module that needs to communicate with the S1000 and F1000 only has to include the .js class and call the respective methods, instead of rewriting them.

Unfortunately JavaScript only supports single class inheritance (Mozilla), which means that it wasn't possible to make a Resource Class with the common attributes, like name, port, etc, and then inherit the S1000 and F1000 classes.

Web Sockets is a new HTML5 API which facilitates the asynchronous communications between Browser and Server by implementing a socket based solution which allows the Server to send data to the client "whenever it needs to" (WebSocket.org). This API comes to fix the hassles of *comet* implementation in Ajax, meaning that techniques like *long polling* wouldn't be necessary. Unfortunately, due to requirement of being Internet Explorer 8 compatible, this technology couldn't be used and *long polling* was chosen instead (). Table 2 displays in green which Browser versions support Web Sockets.

	IE	Firefox	Chrome	Safari
		3.6		
	6.0	9.0 <sub>Moz</sub>		
	7.0	10.0 <sub>Moz</sub>	17.0	
	8.0	11.0	18.0	5.0
Current	9.0	12.0	19.0	5.1
Near future	10.0	13.0	20.0	5.2
Farther future		14.0	21.0	


Table 2 - Web Sockets API browser support ()

## 4.4 The Overview Model

The starting point of the Centralized System's Web Application is the Overview Model, which provides a global view of the system and at the same time also displays what each mxfspeedRail resource is doing in real time. This module is responsible for the system's main monitoring features and its main goal is to inform the user about the resources state.

As most of the other application models, this one was divided in three sections:

- **Individual**, where the user can:
  - View detailed information about task execution progress.
    - As shown in Figure 30, the "Activity" list displays what tasks are being executed at the moment and which ones have completed.
  - Consult event notifications list
    - The "Notifications" list displays the outcome of each task, which can be *completed*, *error* and *canceled*.
  - Check the resource status
    - This information is given by the circle near the resource's name
      - "Disconnected": The centralized system can't reach the resource
      - "Retrieving": The centralized system is trying to connect to the resource
      - "Connected": The centralized system has full access to the resource's services
  - Access each resource interface

- By using the “Jump to” Button the user will open a new window with the resource’s Flex interface
  - View multiple resources simultaneously (Figure 30)
  - Add new resources to the Centralized System (Figure 31)
    - The user can define the resource connection data through the form manually
    - Automatically fill the form by selecting one of the resources found in the network through *Bonjour* and clicking the *Apply* button.
- **Group**, where the user can:
  - Organize resources by groups (Figure 32)
  - View multiple groups simultaneously (Figure 33)
  - View the group progression, which depends on the task progression of each group resource
  - Discover resources with errors, visually indicated by the  symbol.
- **Global** (Figure 34), where the user can:
  - Consult how many resources exist in the System and what type they are (System area)
  - View the last errors from each resource (Errors list)
  - View what kind of operations are being performed (ingest/outgest/capture) and how many resources are executing them (Activity list)

A system administrator can greatly benefit from the use of this new tool. Instead of keeping track of the resources individually, it’s now possible to have a real time understanding of what’s happening in the mxfSpeedRail products as a whole.

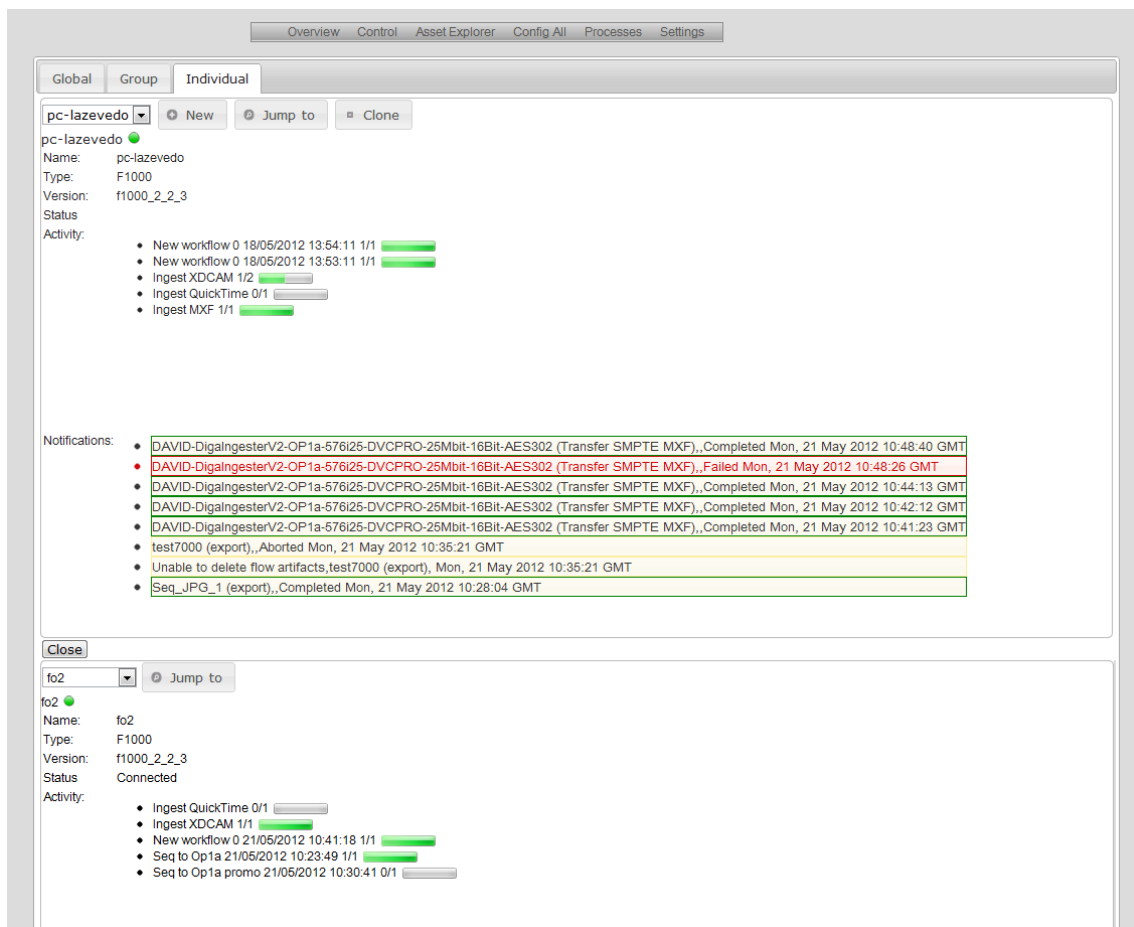


Figure 30 - Overview, Individual Tab with multiple resources



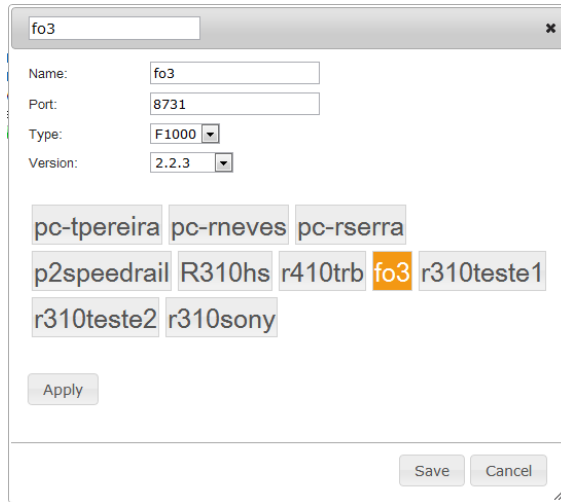


Figure 31 - Add resource dialog

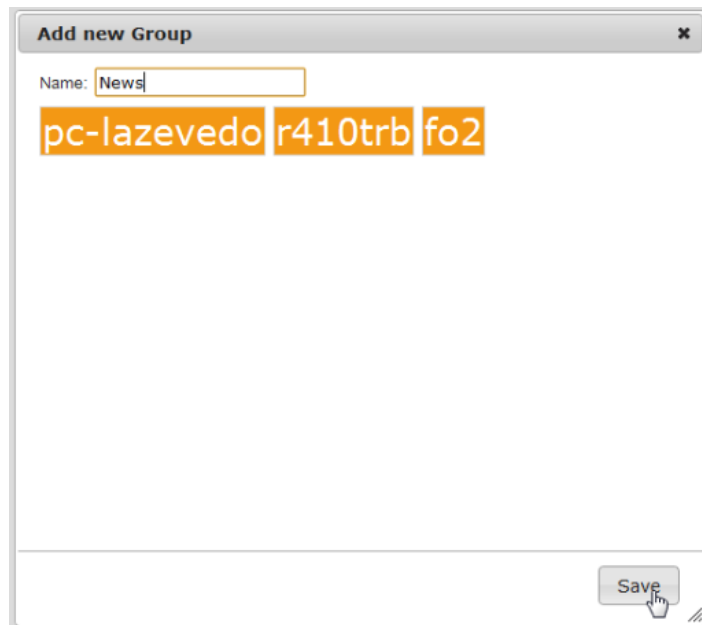


Figure 32 - Organize Resources by Group

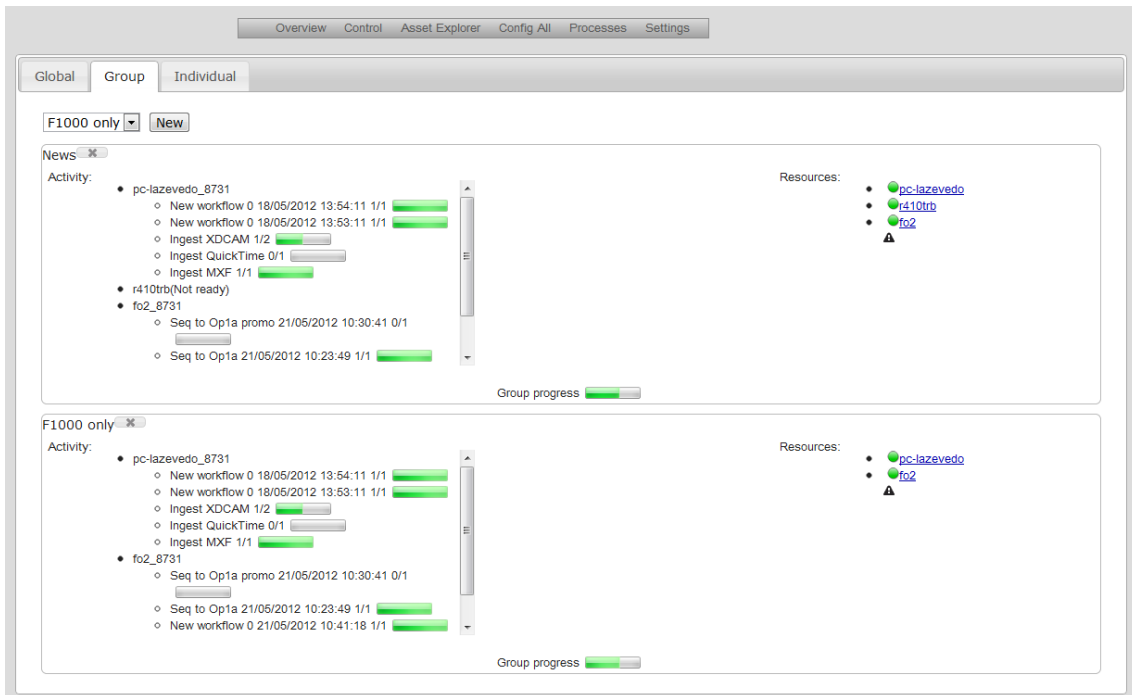


Figure 33- Multiple Groups View

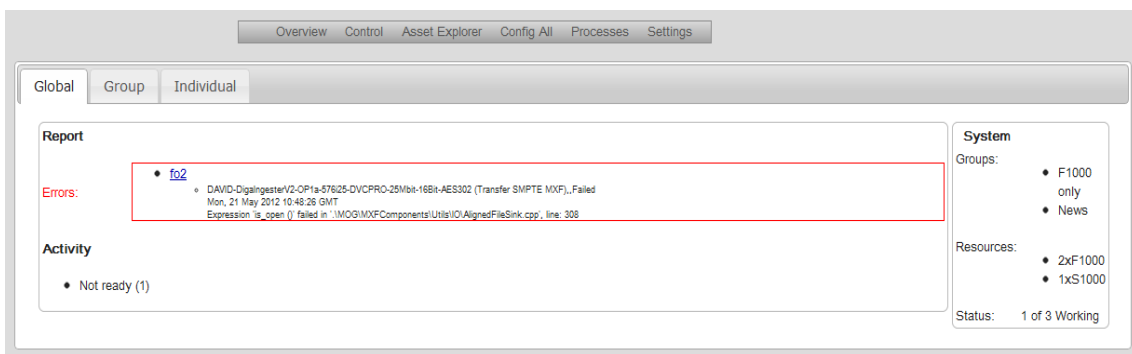


Figure 34 - Global View

To achieve the idealized performance of this model a rather complex solution called Event Hub had to be implemented in the Server.

## 4.5 The Event Hub

Due to the high level of responsiveness demanded by the system, the interface had to be constantly updated in order to display new events fired by the mxfSpeedRail products. This translated in a high number of SOAP requests to keep track of what was happening in the product level, which in turn meant an overburden to the machines. Therefore, the ideal scenario would be the mxfSpeedRail layer warning the Web Server when something relevant happened, instead of constants requests from the server. To achieve this, it was implemented, in the Web Server side, a mechanism called Event Hub Client to which the mxfSpeedRail products were already prepared for, as it is used in the Flex interface. A sequence diagram in Figure 35 shows the difference between the requests sequence with the Event Hub (right) and without it (left). It's obvious that when using the Event Hub the number of requests lowers considerably.

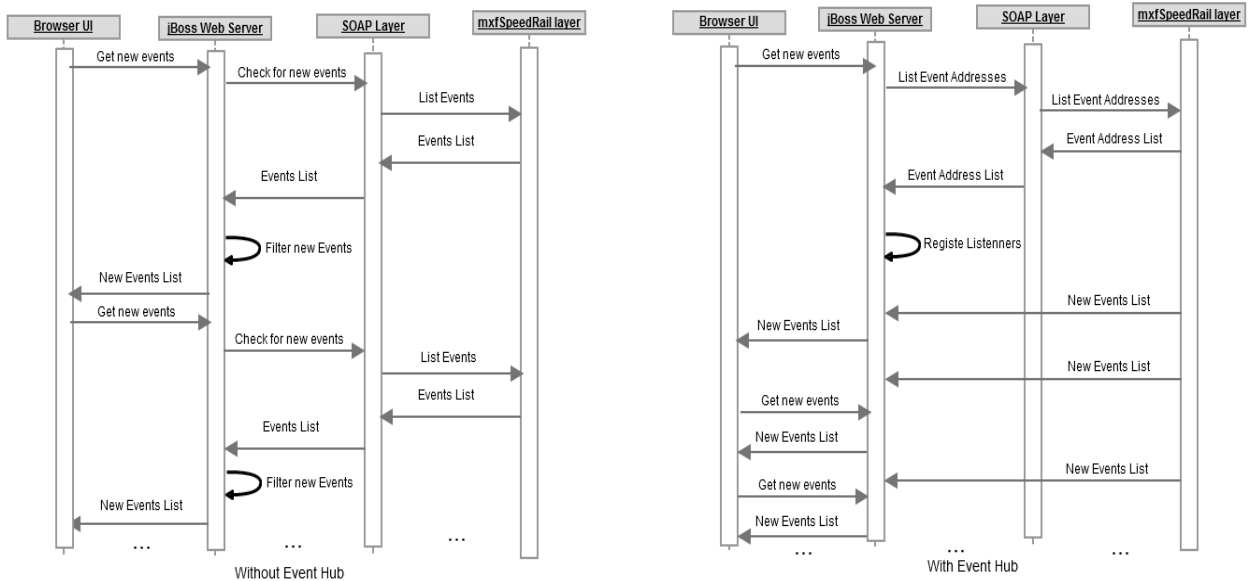


Figure 35 - Event Hub comparison

This feature was accomplished by implementing a *thread* that uses a *socket* which listens for new events returned by the mxSpeedRail products. The Web Server makes a SOAP request in order to know in which addresses the mxSpeedRail products are broadcasting their events list. Then the Web Server launches a new *thread* for each address and opens a *socket*. This *socket* will block until a new event description is sent through the respective address, which will then be processed. The next time the Browser asks if there are new events the Web Server will return the received data.

Implementing this feature on the Web Server was trickier than in the Flex interface, as the server needs to account for multiple mxSpeedRail products while the Flex interface is used for one at a time.

## 4.6 Control

The Control model provides an almost direct interaction with the resources and as the Overview model it is also organized in Global, Group and Individual sections. This serves to give different abstraction levels by providing common commands for all the selected resources. For example, in the Global tab, the user has access to the actions that are supported by all the Centralized System's resources. As shown in Figure 36 there are 4 common commands to all the resource types:

- Shutdown and Restart: which will power off or restart the selected resources
- Download Logs: which will download a zip file from each selected resource containing information about execution history
- Sync settings; which allows to import and export resource settings

On the Groups tab and with the "F1000 only" group selected (Figure 37), an additional option is shown, which is specific to F1000 resources.

The "Sync Settings" command provides a handy feature by allowing the user to choose one source, multiple destinations and synchronize their settings. For example, the user has added two new F1000 to the system and wishes to set them with the same configuration as another one which is already configured. Instead of exporting the settings of the configured one and manually going to the interface of the other two and individually import the file, it's possible to simply identify the "from" and the "to" and click sync (Figure 38). The Centralized System will export the settings from the resource in the "From" list and import to each one of the resources in the "To" list, using the appropriated SOAP methods

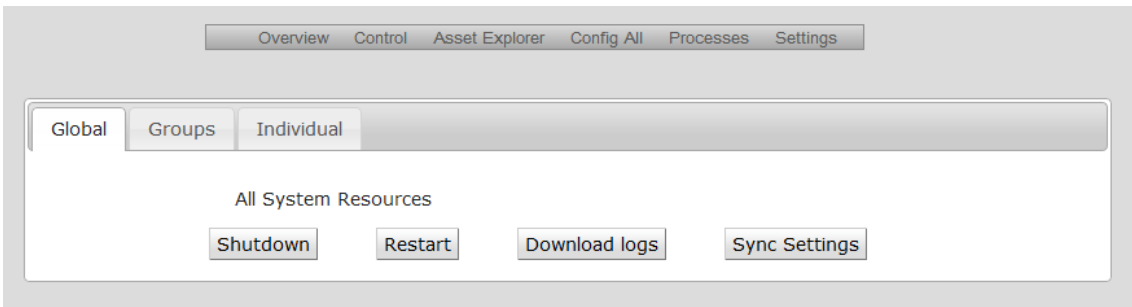


Figure 36 - Possible commands to all resources

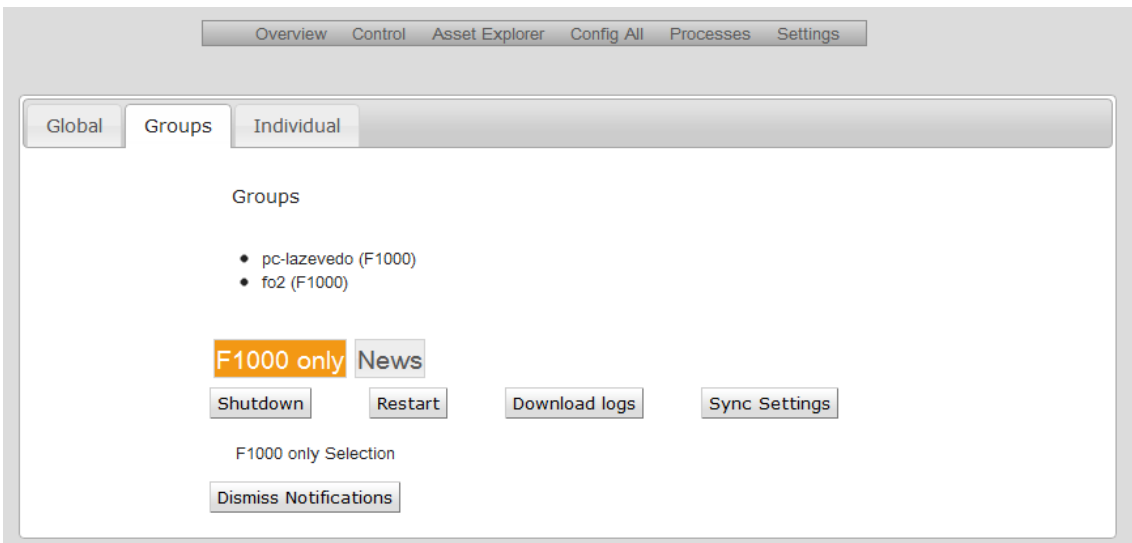


Figure 37 - Control group view

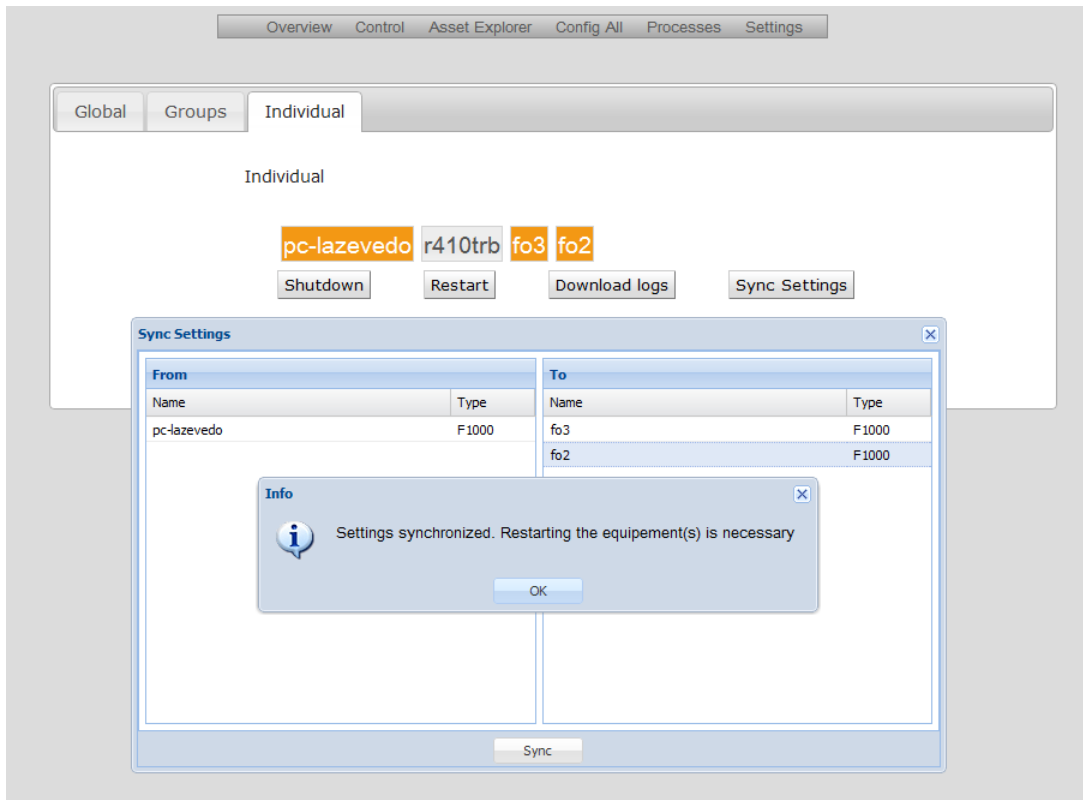


Figure 38 - Synchronizing Settings

## 4.7 Config All

Obviously, configuring the Centralized System’s resources isn’t always as easy as synchronizing settings, sometime the user might want to use the configuration values of a particular resource as the starting point to configure another. “Config All” module provides features to do just that. The user is able to use an existing configuration from one resource and tweak it before applying to another one.

Although there are multiples configuration options (Figure 39) this section will only focus on one to illustrate how this module works.

Before describing in detail the Storage feature, some contextualization is necessary. Storage represents a server where the output of a specific operation, like *ingest*, is going to be stored. What adds complexity to this notion is that the server might be of different types and each type has its own configuration fields. For instance, a storage server of the type “Avid Unity Medianet” needs a Fiber-channel interface address which is absent on other storage types.

The storage configuration procedure is done through the form depicted on Figure 40. The form provides access to all the resources’ storages and when the user chooses one, the fields are automatically set with the corresponding configuration values. Then, it’s possible to replicate the settings to all the resources, to a specific group or to a single product by selecting the Global, the Group or the individual tab respectively. The user can even mix settings values between configurations by “locking” the fields. For instance, to use the same username and password from the *lanshares* storage (Figure 40) to configure a new Storage of the type “Smb Cifs” the user can simply check the *lock* field and the values won’t be changed when the user chooses another existing storage configuration from the list (Figure 41). Obviously it’s possible to create a new set of configuration or edit an existing one by changing the form’s values.

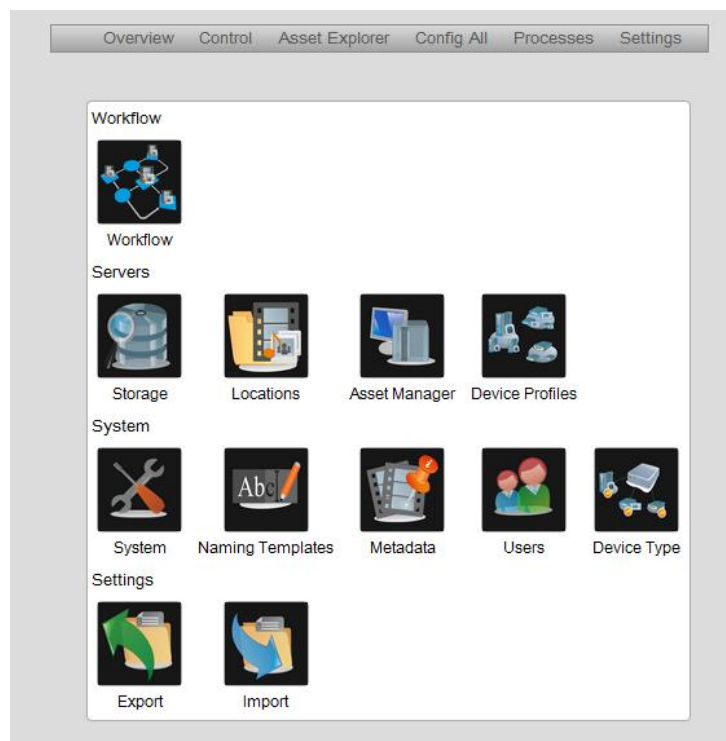


Figure 39 - Config All module

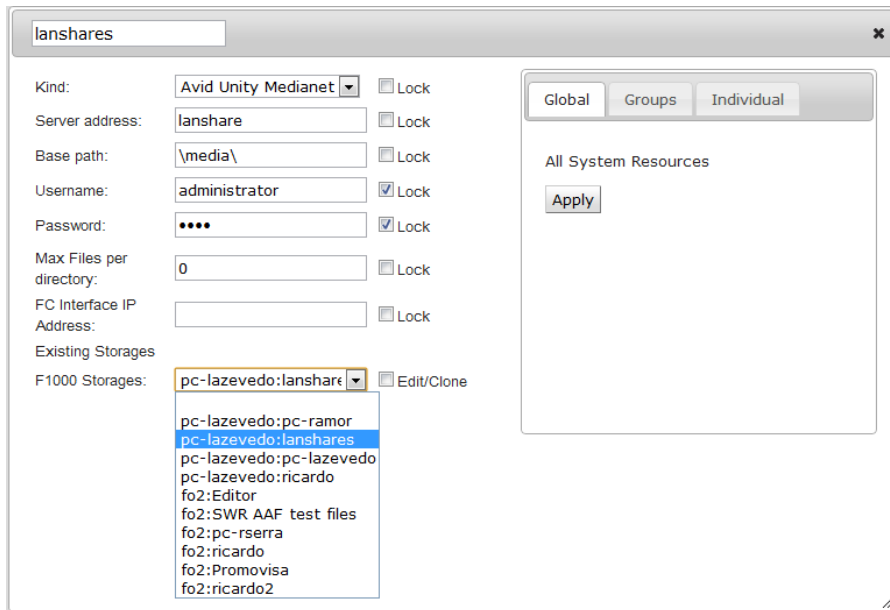


Figure 40 - Storage Form

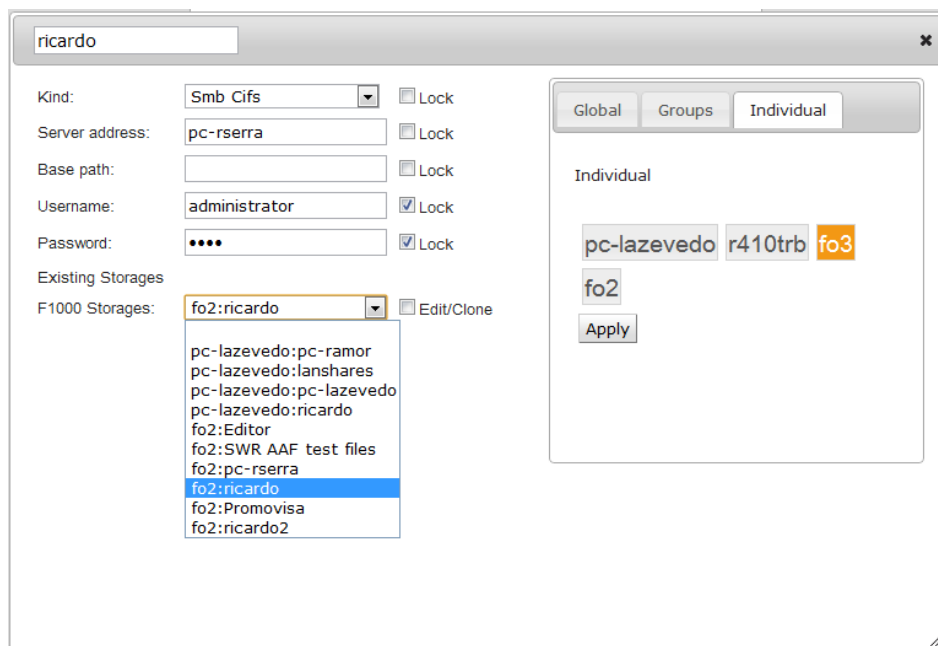


Figure 41 - Locking settings fields

## 4.8 The Settings Module

This module is where the user can define activities and design processes. To actually run the processes the user has to access the “Processes” Module. This division between designing and running was made because each feature is targeted for different users. While the process design is more technical, demanding a basic knowledge of business process modeling using BPMN2.0, running the workflow should be simple to non-technical users. With this module organization the workflow engineers will be responsible for designing the BPMN2.0 process and other users like journalists just need to run it without knowing the subtleties of process design.

To better illustrate the advantages of using a business process based approach to control the resources operations and interactions, an example follows:

A Sports channel is preparing to cover a Football game and the producer wants to save the highlights along with descriptions and then simultaneously send the result to a news channel and to the web. This workflow can easily be described through a business process using the Centralized Solution (Figure 42). A detailed explanation of the diagram follows:

1. The producer describes to the workflow engineer the football coverage sequence
2. The workflow engineer defines the activities to be applied within the business process using the Web Application interface.
3. The defined activities are persisted in the Drools Guvnor.
4. The Workflow Engineer designs the Business Process using the activities retrieved by the jBPM Designer from the Drools Guvnor Repository.
5. Finally, the business process model serialized to BPMN2.0 xml is submitted to the Drools Guvnor Repository

The football sequence is now ready to be used during the event. The next sub section demonstrates how this workflow can be actually implemented using the Centralized Solution.

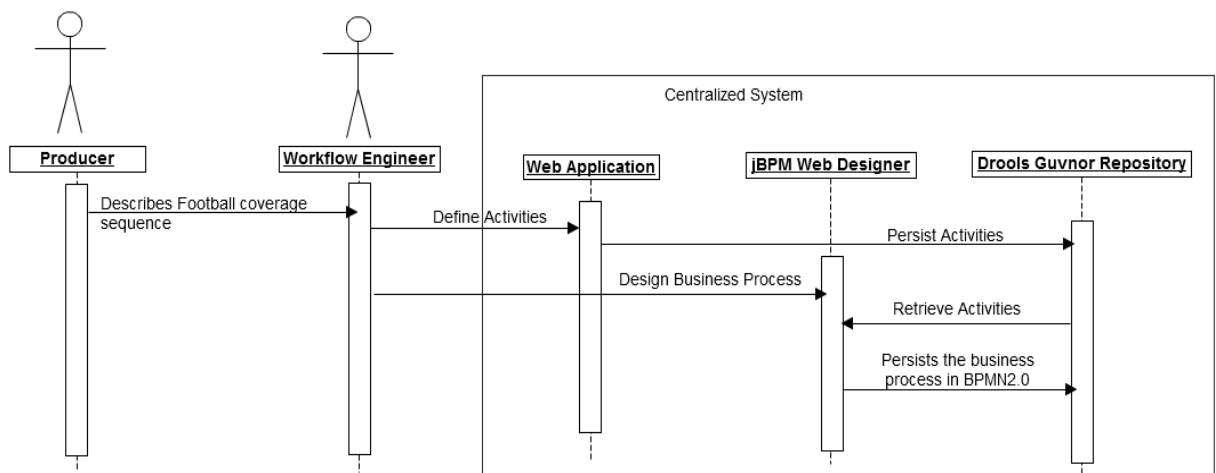


Figure 42 - Football event workflow

#### 4.8.1 Designing a business process

As previously mentioned a process is a sequence of activities. As such it's necessary to first define the activities and then the process.

Activities are added in the "Activity" tab of the "Settings" module (Figure 43). The necessary activities for the football workflow are:

1. **Capture**: during the event will be used to capture the video
2. **Highlights and metadata**: an operator will be editing the captured video to isolate the Highlights and will add the respective descriptions
3. **Ingest Hi-res**: after the highlights and metadata are completed the final result will be ingested to a Hi-resolution version
4. **Ingest proxy**: a lower resolution version of the highlights is processed
5. **Post on web site**: add the proxy video version to the channels web site
6. **Send to news channel**: Send the Hi-resolution version to a news channel

Each one of the activities present in this module is persisted in the Guvnor Repository through a REST interface. By doing this they will be available in the jBPM Designer as Service Tasks.

Service tasks refer to activities that are domain-specific, meaning that they have some sort of specialization related to a domain area. In this case the activities are part of the Centralized System domain and hold attributes to be used with the mxfspeedrail products.

The form present in Figure 43 is used to configure the *ingest* settings to be used by the F1000.

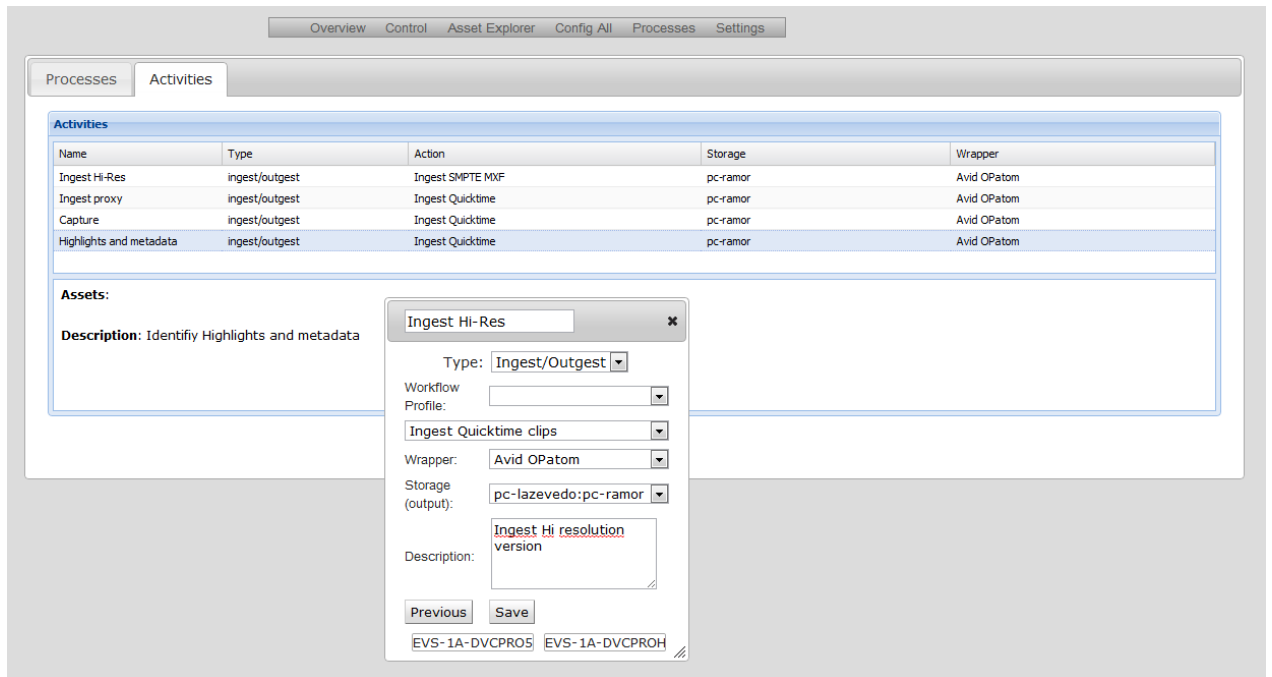


Figure 43 - Activities

After setting the activities, the workflow engineer needs to design the process, which is done in the "Processes" tab.

As shown in Figure 44, the Processes tab displays a list of already designed processes, like the News Workflow, and provides the user a preview so it's not necessary to open jBPM Designer to view the activities. The user also has the option to "Add" a new business process or "Delete" an existing one.

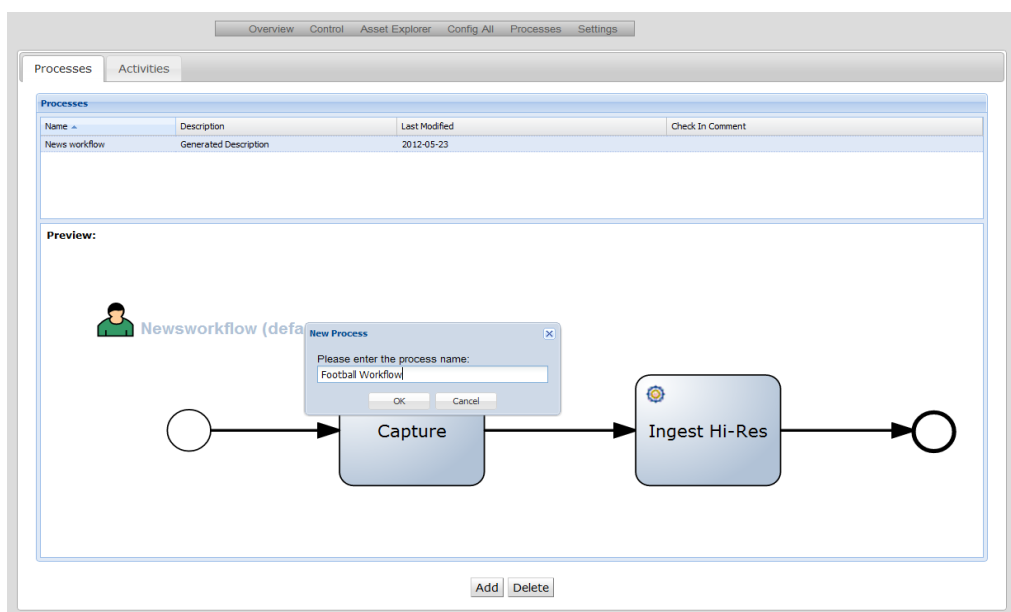


Figure 44 - Settings Processes Tab



To “Add” a business process the user must specify a name and afterwards the jBPM Standalone Web Designer opens in a new window.

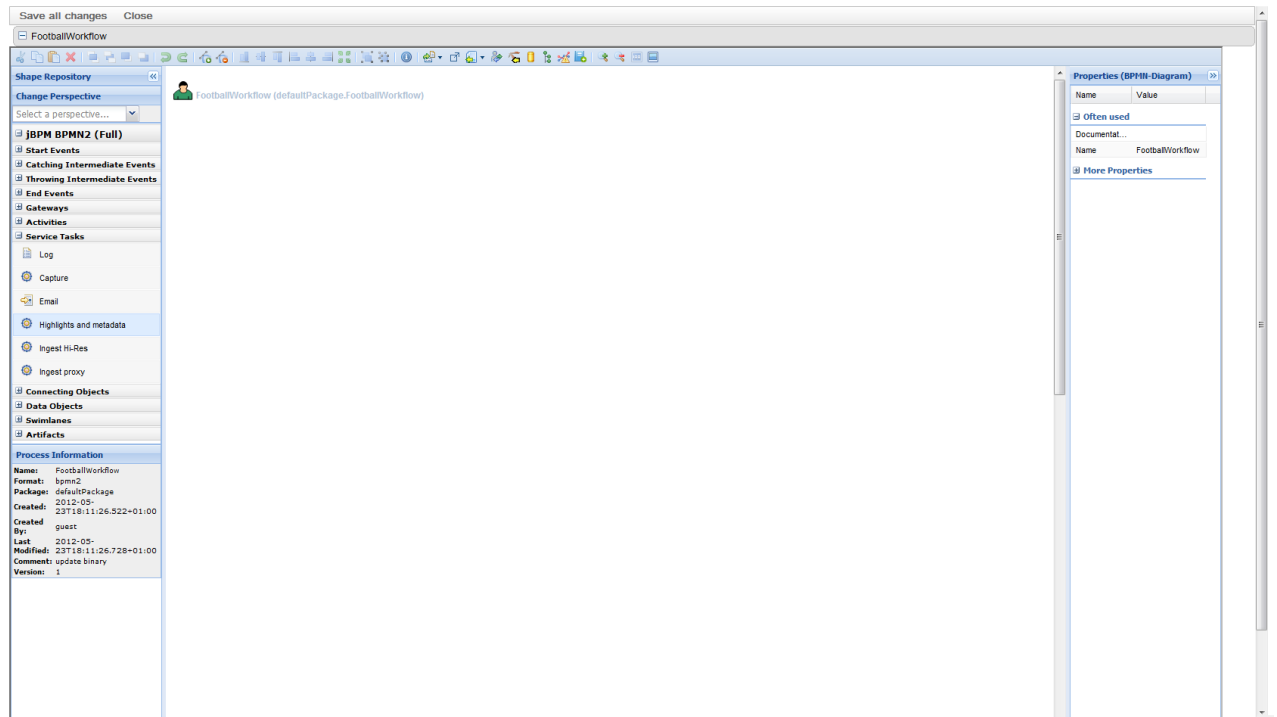







Figure 45 - jBPM Standalone Web Designer

The previously defined activities are available under “Service Tasks”, on the left, and the user can apply them to the workflow. The final business process is visible on Figure 46.

All BPMN2.0 business processes need a starting point which has one output and zero input. The start event  defines the beginning of the workflow and has a corresponding end event , which is also required. The first two activities of the business process are capturing the game and simultaneously isolate the highlights with metadata. To achieve this parallelism a BPMN2.0 component called “parallel gateway”  is used, which forks the workflow in two branches: the “Capture” and “Highlights and Metadata” activities. While the first is a “Service Task”, which should be automatically handled by the Centralized System in conjunction with the S1000 product, the second will be performed by a user. This distinction between both activities is visually identified by the  and  icons respectively. Only when both activities have finished, will the workflow proceed. This condition is forced by the second parallel gateway. The sequence will then be divided again in two branches, one to *ingest* the commented highlights into a Hi-res file and send it to a news channel and the second where a lower version of the video is made and posted on the Web site. As before, the activities that are domain-specific to the Centralized System and the mxSpeedRail products are identified accordingly, differentiating themselves visually from the other ones. Because the two last activities don’t involve the mxSpeedRails products and might be a mixed of manual and automatic interaction they are general activities and have no associated icon. The great advantage of using BPMN2.0 is that all these graphic components are standards, which means that anyone who knows the modeling language understand the business process.

After finishing the design, the user can save it to the drools guvnor repository under version control along with a comment. Right before the jBPM Web Designer sends the serialized BPMN2.0 (Figure 47) to the repository, a SVG representation of the business process is sent to the Web Application and is associated with the respective workflow. By doing this, it’s possible to provide the preview illustrated in Figure 48.

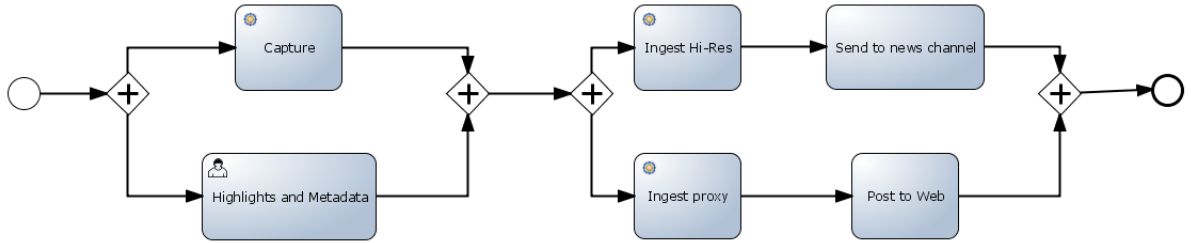


Figure 46 - Football coverage event business process

```

BPMN2 Source
1 <?xml version="1.0" encoding="UTF-8"?>
2 <bpmm2:definitions xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
3   xmlns:bpmm2="http://www.omg.org/spec/BPMN/20100524/MODEL"
4   xmlns:bpmmdi="http://www.omg.org/spec/BPMN/20100524/DI"
5   xmlns:dc="http://www.omg.org/spec/DD/20100524/DC"
6   xmlns:di="http://www.omg.org/spec/DD/20100524/DI"
7   xmlns:drools="http://www.jboss.org/drools" id="nc5aYKWHHeGXIt_qYQeM_g"
8   xsi:schemaLocation="http://www.omg.org/spec/BPMN/20100524/MODEL BPMN20.xsd"
9   targetNamespace="http://www.omg.org/bpmm20">
10  <bpmm2:process id="defaultPackage.FootballWorkflow"
11    drools:packageName="defaultPackage" name="FootballWorkflow" isExecutable="true">
12    <bpmm2:parallelGateway id="B11BBA9E-86EF-4EA6-A506-DOC8BC4BDBBD"
13      drools:bgcolor="#####" name="M gatewayDirection="Diverging">
14      <bpmm2:incoming_18505017-46B7-44F7-86D9-B8EF62C392DB</bpmm2:incoming>
15      <bpmm2:outgoing_00700C3E-EF8B-474F-AB8E-E4527F20949C</bpmm2:outgoing>
16      <bpmm2:outgoing_CD01EF01-E7FC-452D-A3E6-599EF7ED799D</bpmm2:outgoing>
17      </bpmm2:parallelGateway>
18      <bpmm2:startEvent id="_2DE7D687-20D6-4942-9B4E-0FD47FAAFD4D"
19        drools:bgcolor="#####" name="">
20        <bpmm2:outgoing_18505017-46B7-44F7-86D9-B8EF62C392DB</bpmm2:outgoing>
21        <bpmm2:startEvent>
22        <bpmm2:sequenceFlow id="18505017-46B7-44F7-86D9-B8EF62C392DB" sourceRef="_2DE7D687-
23          20D6-4942-9B4E-0FD47FAAFD4D" targetRef="B11BBA9E-86EF-4EA6-A506-DOC8BC4BDBBD"/>
24        <bpmm2:userTask id="8D5F63C3-35E1-4270-96B2-5C46D8FF66B8"
25          drools:scriptFormat="http://www.java.com/java" name="Highlights and Metadata">
26          <bpmm2:incoming_00700C3E-EF8B-474F-AB8E-E4527F20949C</bpmm2:incoming>
27          <bpmm2:outgoing_COE35E44-2B69-4F06-B644-DSA21D0E1191</bpmm2:outgoing>
28          </bpmm2:userTask>
29          <bpmm2:sequenceFlow id="00700C3E-EF8B-474F-AB8E-E4527F20949C" sourceRef="B11BBA9E-
30            86EF-4EA6-A506-DOC8BC4BDBBD" targetRef="8D5F63C3-35E1-4270-96B2-5C46D8FF66B8"/>
31          <bpmm2:task id="_62C35DEB-46FB-4647-89DA-BD8D7670217C" drools:taskName="Capture"
32            name="Capture">
33            <bpmm2:incoming_CD01EF01-E7FC-452D-A3E6-599EF7ED799D</bpmm2:incoming>
34            <bpmm2:outgoing_9C6CEE7D-9BE3-4DCF-837E-2F95C1428DD9</bpmm2:outgoing>
35            <bpmm2:ioSpecification id="nc6BcqWHHeGXIt_qYQeM_g">
36            <bpmm2:dataInput id="_62C35DEB-46FB-4647-89DA-BD8D7670217C_TaskNameInput"
37              name="TaskName"/>
38            <bpmm2:inputSet id="nc6BcaWHHeGXIt_qYQeM_g"/>
39            <bpmm2:outputSet id="nc6BcqWHHeGXIt_qYQeM_g"/>
40          </bpmm2:task>
41        </bpmm2:sequenceFlow>
42      </bpmm2:parallelGateway>
43      <bpmm2:outgoing_9C6CEE7D-9BE3-4DCF-837E-2F95C1428DD9</bpmm2:outgoing>
44      <bpmm2:ioSpecification id="nc6BcqWHHeGXIt_qYQeM_g">
45      <bpmm2:dataInput id="_62C35DEB-46FB-4647-89DA-BD8D7670217C_TaskNameInput"
46        name="TaskName"/>
47      <bpmm2:inputSet id="nc6BcaWHHeGXIt_qYQeM_g"/>
48      <bpmm2:outputSet id="nc6BcqWHHeGXIt_qYQeM_g"/>
49    </bpmm2:parallelGateway>
50  </bpmm2:process>

```

Figure 47 - BPMN2.0 source

Figure 48 - Football event workflow preview

## 4.9 The Processes Module

The Processes Module is responsible for running the designed business processes by interacting with the jBPM API. A similar list to the one shown in the Settings module is visible in the processes tab but has now a “Run” option (Figure 49).

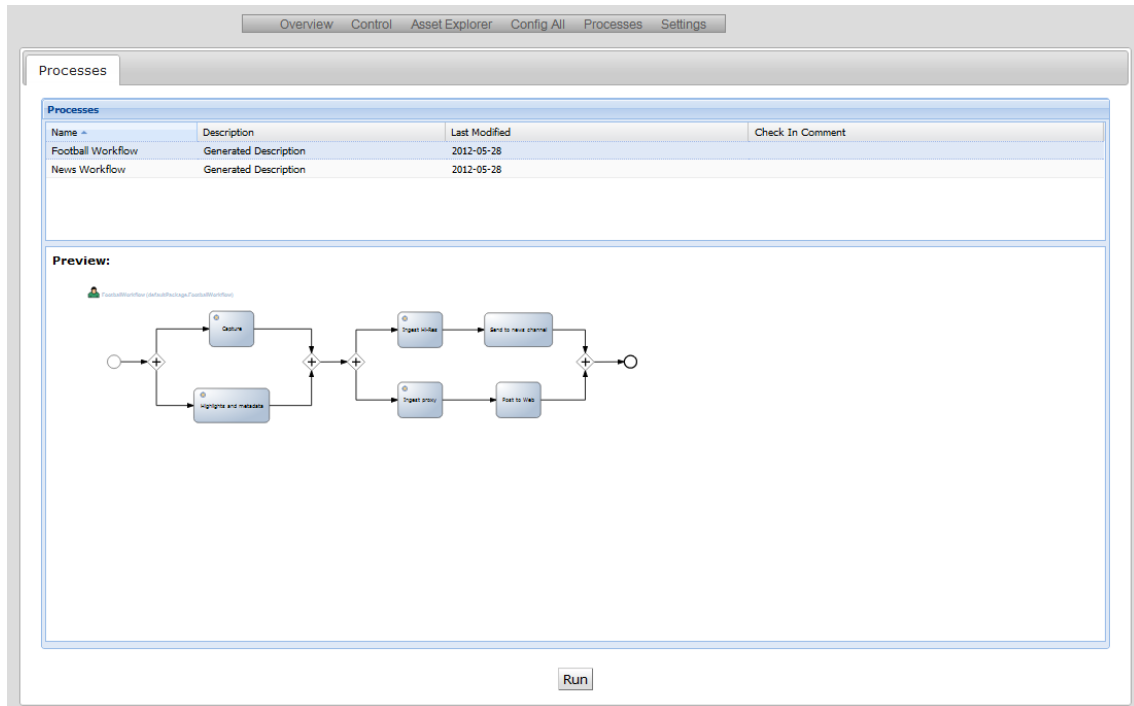


Figure 49 - Processes run

The workflow design and execution features were separated so users from different areas, like the producer and workflow engineer, have well defined roles. The producer doesn't need to know how to design a business process but should be able to easily run it when necessary. On the other hand, the workflow engineer is only responsible for implementing the BPMN2.0 description. By keeping the modules separated this organization is forced on the users. The implementation approach even allows a TV channel to outsource the workflows design as the Drools Guvnor repository where the BPMN2.0 xml's are stored can be accessed from the internet.

Although the F1000 is able to perform up to 3 tasks in parallel, the workflow will be completed faster if the Centralized System balances the resources' workload. Because the system has access to what the resources are doing in real time it should be “smart” enough not to overburden an already busy resource with new tasks. As such a workload balancing algorithm was implemented. When the user runs the business process, the Centralized System assigns the tasks to the resources which have the least running operations. For instance, in the example shown in Figure 49 the Web Server asks each F1000 how many uncompleted operations they have and orders them accordingly. The least busy gets the first task, the second least busy gets the other task and so on. All of this balancing is transparent to the user, who doesn't even need to know which mxfSpeedRail is able to perform an *ingest* operation.

## 4.9.1 How processes are run with jBPM

This sub-section explains how the Centralized System makes the connection between the jBPM API and the mxfspeedRail SOAP layer in order to run the business process.

jBPM API has three main components:

- The knowledge base, which is used to load the business process model. This involves parsing and validating the BPMN2.0 asset.
- The session, which provides methods to interact with the engine, for instance, starting the business process.
- The *events*, which is nothing more than an interface with methods like *afterProcessCompleted*.

When the user issues the run command, an AJAX request is sent to the Server with the selected process name and id. On the server side this information is used to create a URL pointing to the BPMN2.0 file saved on Drools Guvnor repository. Then the following implementation sequence is used:

1. The knowledge base is instantiated using the process model from the URL.
2. A *StatefulKnowledgeSession* instance is returned from the knowledge base so it's possible to interact with the workflow engine.
3. In order for the workflow engine to know what to do when the execution sequence reaches the *ingest* operations it's necessary to register *WorkItemHandlers*. *WorkItemHandler* are classes which implement the methods *executeWorkItem* and *abortWorkItem*. These classes hold the logic to communicate with the mxfspeedRail products during Service Tasks execution.
4. Finally the business process is started by invoking *statefulKnowledgeSession.start(processID)*. The *processId* is necessary because a Session might have more than one process from the knowledge base.
5. Whenever the workflow engine finds a Service Task it will call the corresponding *WorkItemHandler*, which in turn invokes the associated SOAP method.

The described sequence is illustrated in Figure 50.

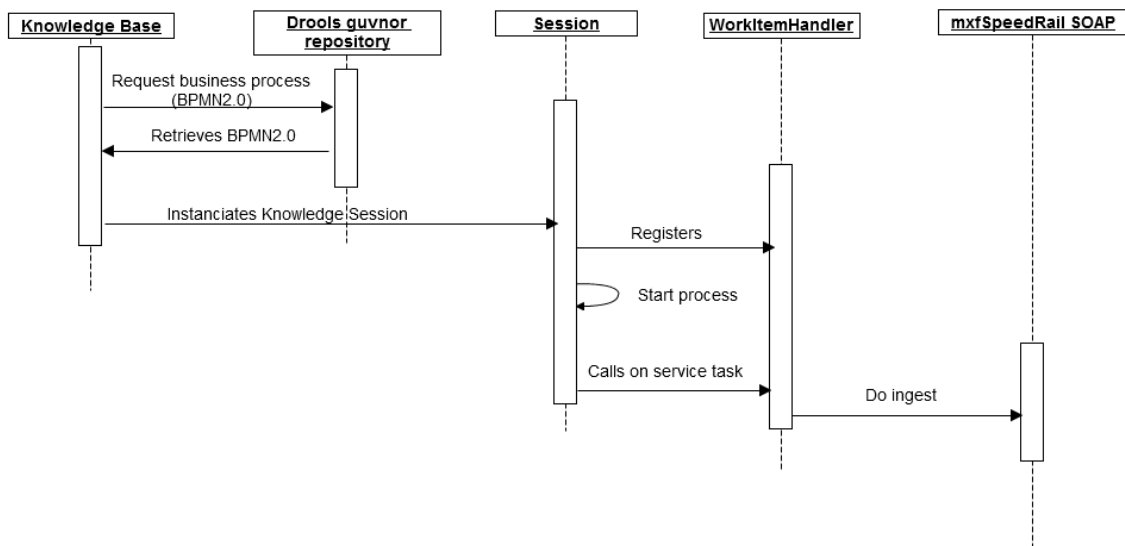


Figure 50 - jBPM/SOAP interaction

## 4.10 Asset Explorer

Assets are basically clips saved on locations and are used as inputs for the *ingest* and *outgest* operations. Obviously things get more complicated when an Asset is a group of files with one of them serving as a pointer to the others, but to keep things simple they are referred as single files during this section.

A location is a folder which has an Asset Type associated and contains the respective Assets. Finding a specific asset within all the locations configured in a single mxfspeedRail product is quite easy, the user only needs to select the locations to search in and write a text representing the asset name or field on metadata. What isn't as trivial is searching for a file in all the locations of all the mxfspeedRail products. The user would have to access each product interface, select all the locations and search for the asset. This operation gets even harder with the increase of resources and it's common for locations to be scattered along different products. The centralized Solution remedies this problem by allowing searching for an asset on multiple locations even if they are configured in different mxfspeedRail products. This feature is implemented on the Asset Explorer Module.

The module provides a tree view of the resources and respective locations (Figure 51). By selecting a single location the user is able to access all the assets in that location. The real advantage is the freedom this module provides by allowing the user to select multiple locations and joins the results together, even if the locations are from distinct resources. The user can easily view all the assets on all the resources by selecting the "folder like" icons which represent a mxfspeedRail product. In Figure 52, the user is searching on multiple resources for assets which have the word David on the name or metadata field. The filtering is done in real time through Ajax.

Another handy feature is the possibility to exchange locations between mxfspeedRail products. For example, the user wants to *ingest* a specific clip with the F1000 (fo2) but doesn't know where the asset is. With the asset explorer the user only needs to search for the file and drag and drop the corresponding location to the fo2 folder (Figure 53), then the resource is properly configured to use the clip.

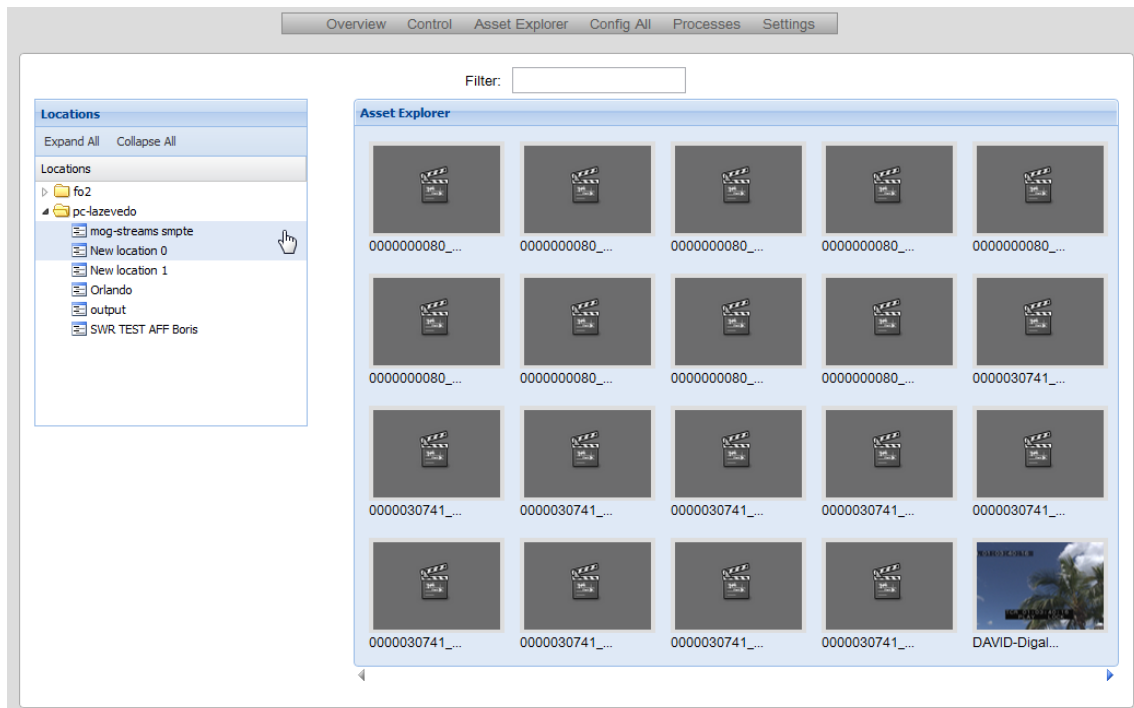


Figure 51 - Asset Explorer multiple locations

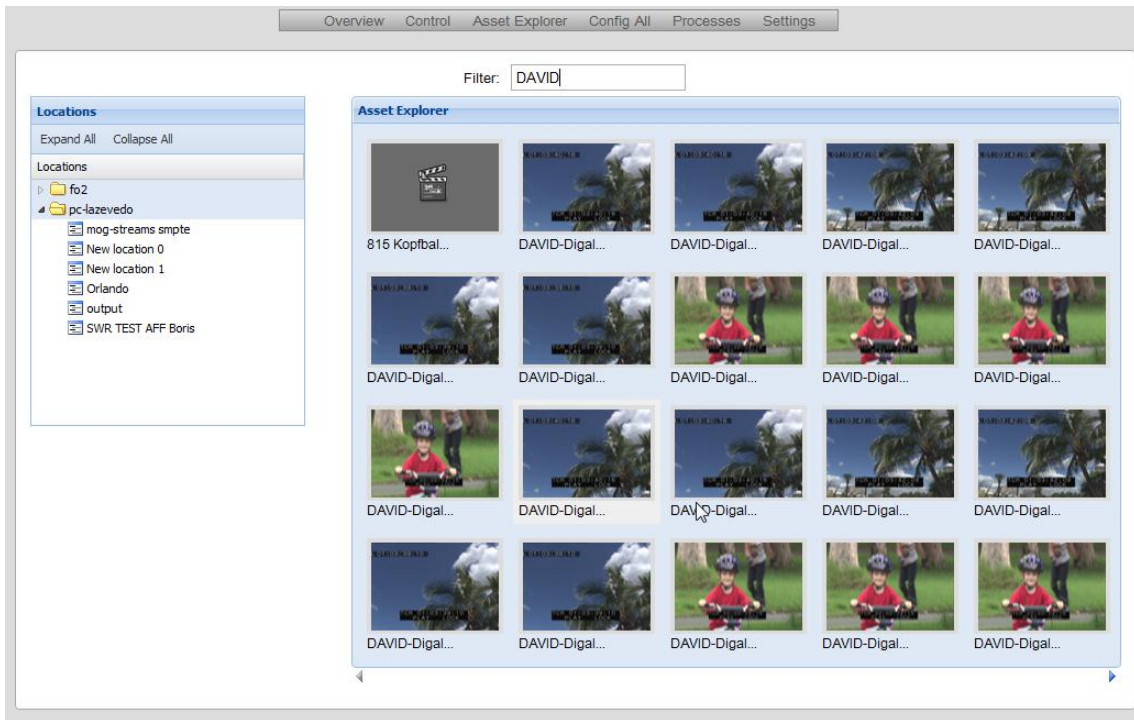


Figure 52 - Real time asset filter by name and metadata field

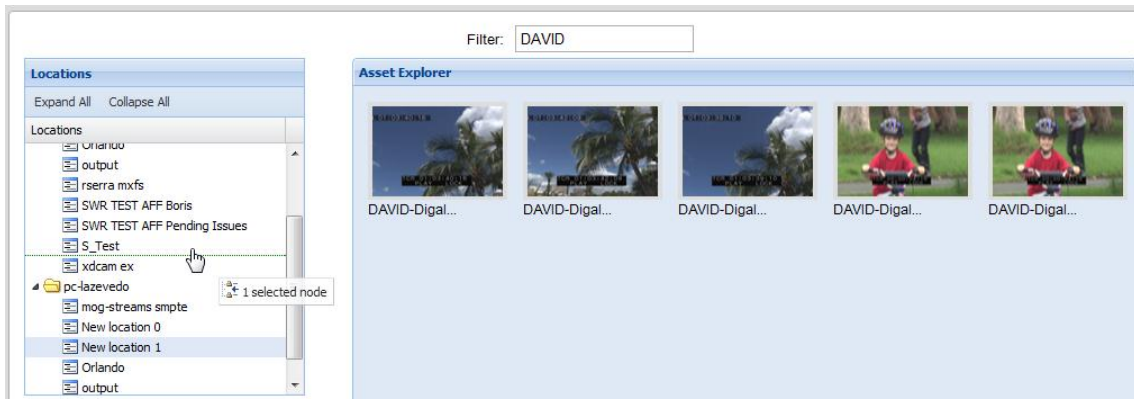


Figure 53 - Adding "New Location 1" to fo2 by drag and drop

## 4.11 Tests and Results

This section describes the outcome of the load balancing algorithm, how complex workflows are handled, the integration between process execution and the Asset Manager and describes the testing approach.

While there are numerous workflows possibilities, this section will focus on a single example that covers the main difficulties of task assignment.

The workflow depicted on Figure 54 is composed by two parallel tasks which should be done simultaneously and a third which uses the combined output of the first pair. This means that the user must specify the input for the initial tasks while the last activity automatically uses all the Assets from the previous *ingest* operations.

In order to run the process the user clicks the “Run” button and is presented with a message informing that assets are missing for the “QuickTime to OP1a” activity. Then the user is able to select the respective clips through the Asset Explorer (Figure 56). The user has now access to all the assets in any of the added resources. After selecting all the desired assets (Figure 57) and pressing “Done” the same process is repeated for the other activity “MXF to OP1a”. When all the assets are chosen jBPM starts the process execution. When it reaches a task it queries the system in order to find which of the resources has less workload and assigns the activity to it. In this test only two resources were being used and both were free, as such the first parallel activity is assigned to a F1000 called “pc-lazevedo” and the second to a F1000 called “fo2”. Only after the two initial tasks are completed will the jBpm proceed to the final one. The third activity will use as input the output from the previous tasks. This is proved in Figure 58 where “pc-lazevedo” finished one of the first tasks “Ingest MXF to Op1a”, while the second was assigned to “fo2”. Finally the third activity “Op1a to Avid” was performed by the F1000 “pc-lazevedo” which was free at the time.

Instead of assigning all the operations to one resource, the work was balanced to the two mxfSpeedRail products and assets were “exchanged” between resources and all of this was done in a way transparent to the user.

Notifying the user about events that happened in the lower SOAP level wasn’t trivial. The user had to be informed when a workflow was completed (Figure 59) which could take minutes or even hours and http protocol does not allow “pushing” data from the server to the client (Google). How this “inconvenient” was handled is detailed in the next section.

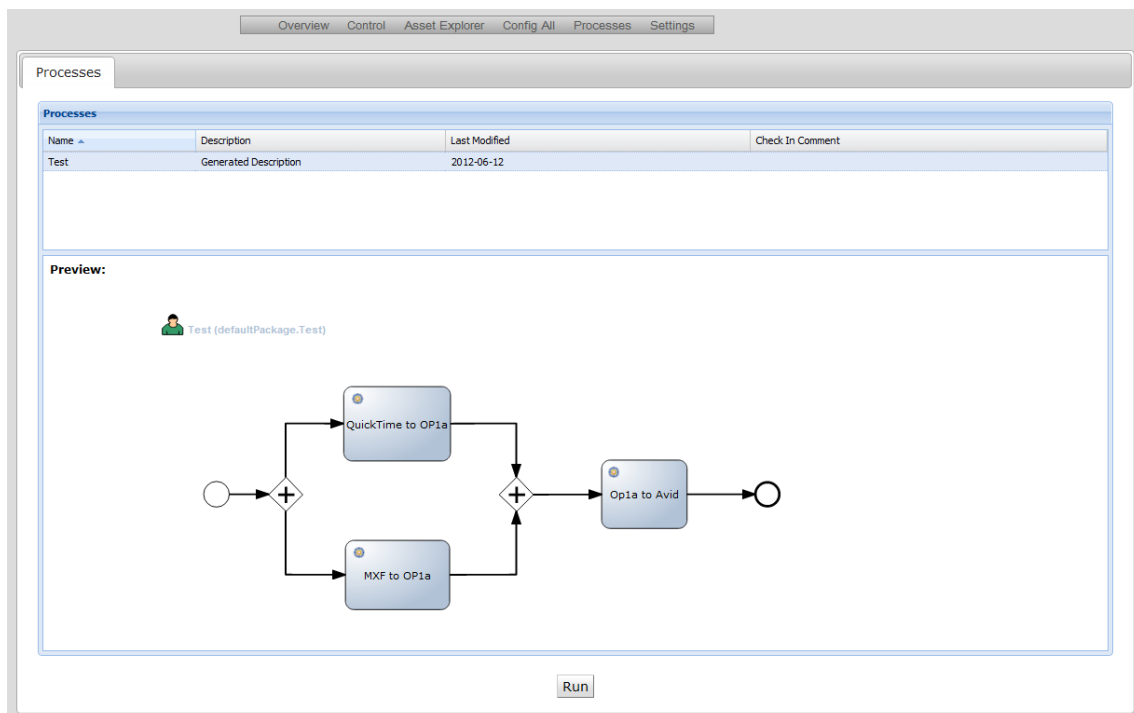


Figure 54 - Test Workflow

age.r estj

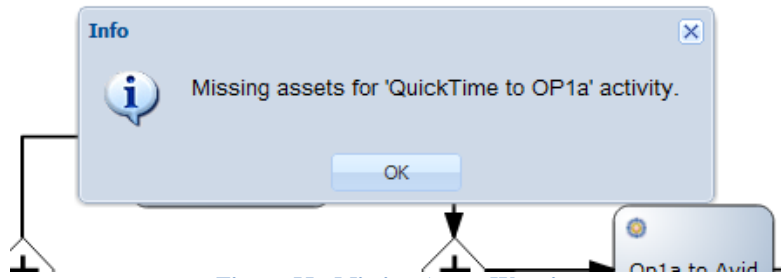


Figure 55 - Missing Assets Warning

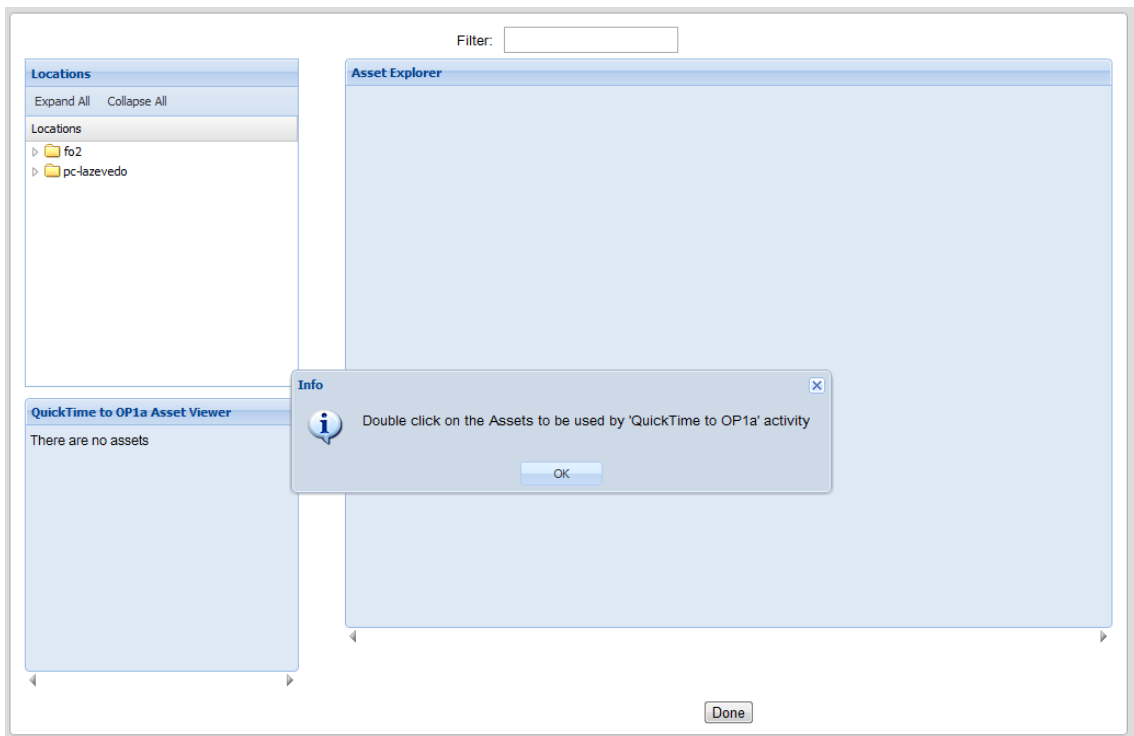


Figure 56 - Asset Explorer and Workflow Interaction



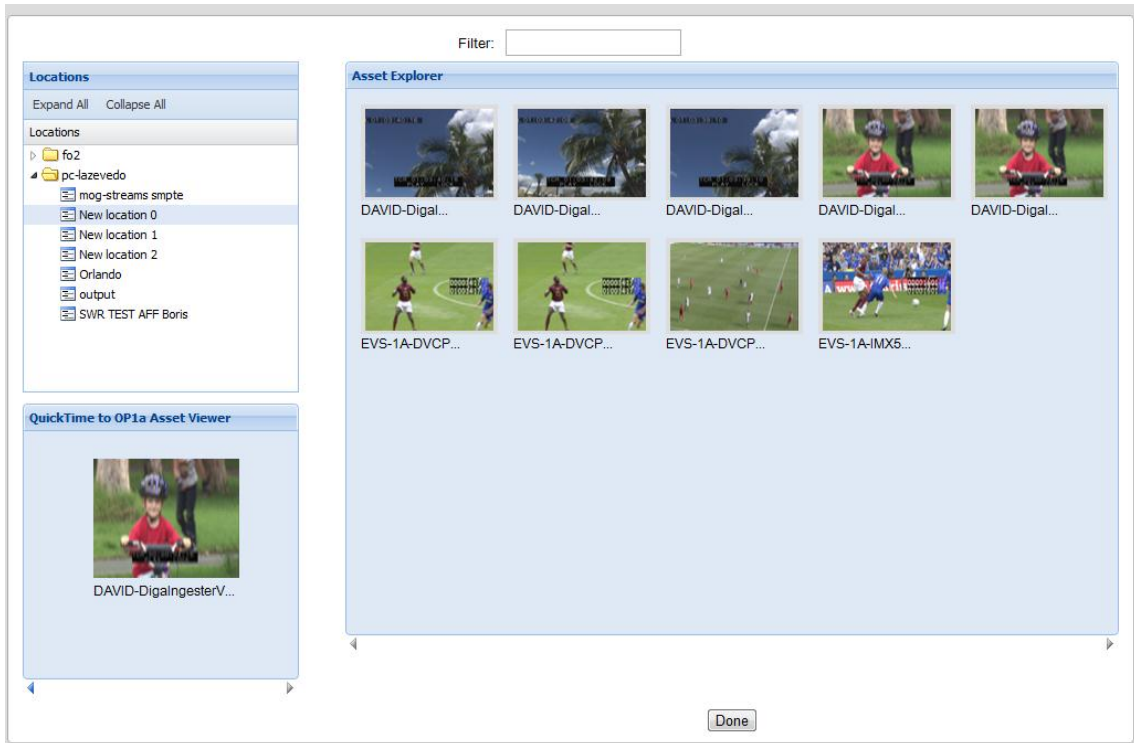


Figure 57 - Selected Assets (left down corner) for Activity

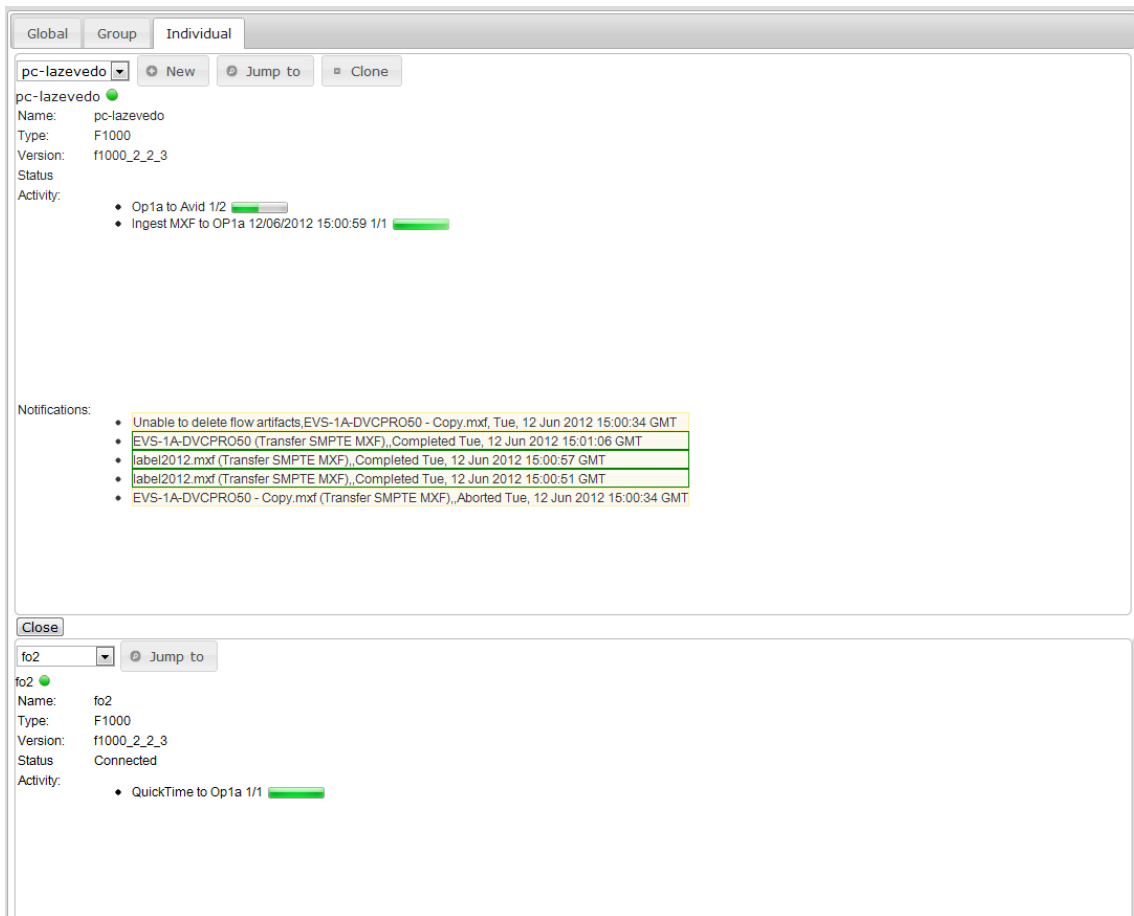


Figure 58 - Test Result

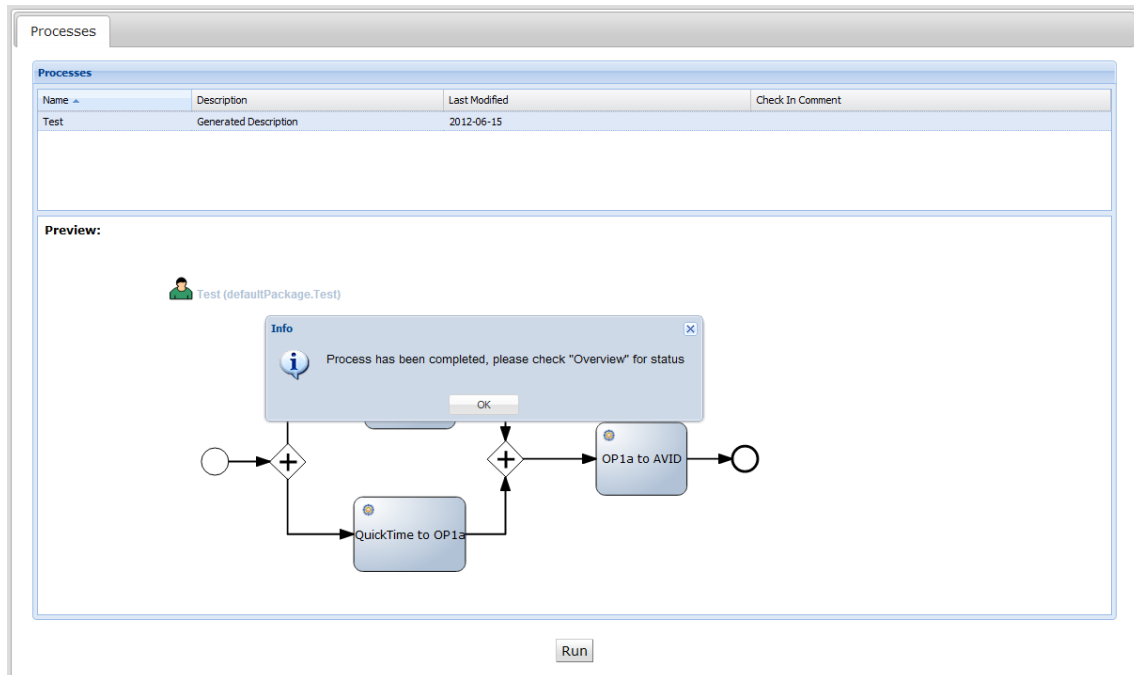


Figure 59 - Workflow Completed Notification

In order to validate the application with the system requirements and to review if the features were correctly implemented, tests were performed at the end of the iterations. The majority of the testing was done manually because most of the Centralized System operations affected the mxfspeedRail products. This meant that unless unit testing was implemented in the mxfspeedRail .NET layer, the only way to efficiently validate the results was by analyzing the outcomes through the interfaces.

## 4.12 Difficulties

Due to the Centralized System and mxfspeedRail complexity there were some implementation challenges. The following three were the most difficult ones:

### 4.12.1 jBPM Standalone Web Designer

The jBPM API and the Drools Guvnor repository were relatively easy to integrate with the Web Application but the Standalone version of the jBPM Web Designer was fundamentally broken and required several *hacks* to function properly.

The basic usage of the Standalone designer is to open a BPMN2.0 process in a new window so it is embedded in the caller application. The parameter required by the designer is an *AssetID* which uniquely identifies the asset to be loaded from the repository. The problem is that when the designer opens the BPMN2.0 process, it automatically changes the URL which makes impossible to refresh the window. Another problem was related to consecutive saves through the “save all changes” button. The Web Designer needs to be refreshed after each save otherwise an error related to check in authorization. Finally the “close” button also didn’t work.

To resolve these issues the Standalone Web Designer was loaded within an *iframe* and because the Web App was served in the same domain and port as the designer, there wasn’t “Cross domain” restriction, which allowed adding event handlers to the buttons. The final solution was a pair of JavaScript functions which implemented the refresh call in the “save all buttons” and the close operation in the “close” button.

Taking an image capture of the business process to show it as a preview also wasn't trivial due to the Web Designer JavaScript API not being documented and required more code injection.

### 4.12.2 The Event Hub

As previously referred, developing the Event Hub wasn't trivial and although its objective was already described, the implementation subtleties weren't.

The Event Hub listens for events fired from the mxfSpeedRail products and makes them available to the Web interface. These events can be of several types and come from multiple mxfSpeedRail products. To properly handle them, it was required to have a *thread* for each resource and use a mapping system to identify what to do with a certain event. This mapping was done by implementing events on the server side. When an event of type *x* was listened by the corresponding Event Hub *thread*, a handling event of type *y* was triggered on the Server. Java doesn't have a direct notion of events like Flash or C# but it has *interfaces* which can have multiple implementations. Thus, a class implements a *CallBack* interface and registers itself on map *hashed* by event types. When the *thread* receives an event, it only needs to call the *run* method of the corresponding interface. *eventHandlerMap[eventType].run()*. The method implementation depends on the class associated to that event type.

### 4.12.3 Synchronizing Settings between mxfSpeedRail

In general, most of the exposed features of the mxfSpeedRail products behaved fairly well when used in a Centralized Environment, but there was an exception related to the settings synchronization.

Exporting settings from a mxfSpeedRail product and then importing to another is done through the Flex interface and involves downloading and uploading a zip file. To automate this operation the Centralized System had to download the zip file from an URL and then simulate a Browser upload to multiple mxfSpeedRail resources. This meant low level XML manipulation and because even a character would invalidate the parsing operation it took more time than expected.

### 4.12.4 “Bubbling” Events

Due to http's request/response unidirectional nature it's not easy to “bubble up” events from the SOAP layer to the Web Browser (Google).

When a SOAP request for a long running operation, like the ingest tasks described in the test section, is made, the method returns instantly and monitoring its progression required the use of the Event Hub. A similar difficulty happens with notifying the GUI (Web Browser) about events like workflow completion which might occur minutes or even hours after invocation. This required implementing an event registration architecture that allows GUI instances to monitor events from specific workflows. In order to minimize the performance “hit” of constant pooling, which involves regular queries from the GUI to the Web Server and consequently to the mxfSpeedRail products, a rather complex thread synchronization scheme was used.

When the user runs a Workflow the Web Browser makes an Ajax request to the web server which in turn starts the business process execution and returns a *processId*. After receiving the response the GUI makes a “monitoring” request using the *processId* and an *eventCounter*, which informs the server about the last event received by the client instance. Upon receiving this request the server sends an *Object* instance to a *Map* which associates a *processId* to *monitors*. Right after, the server blocks the *servlet* thread by calling *.wait()* on the

*Object* instance. While this happens the workflow is being run asynchronously by *threads*. Whenever an operation is completed by the mxfSpeedRail products an event is sent to the respective listening *socket* on the server, which in turn will unblock and call *.notify()* on the process monitor using the *processId*. This will unblock the *servlet* instance, which was waiting on the corresponding *monitor Object* instance and in turn sends to the Web Browser the workflow event.

Figure 60 depicts the events architecture which is compliant with FIMS guidelines (AMWA-EBU, 2011).

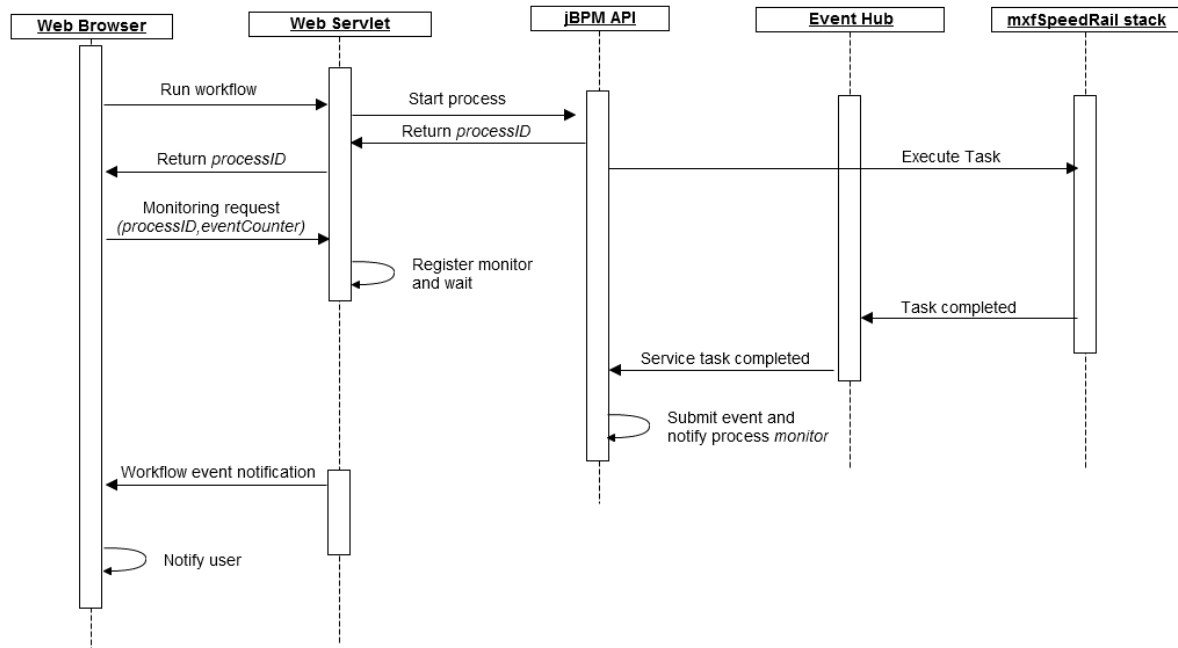


Figure 60 - Events Architecture

## 4.13 Conclusions

This chapter covered the implementation process of the web application and showed the main functionalities of the Centralized System. The iteration methodology fit quite well with the modular approach during development, resulting in a flexible structure where features were easily added. The requirements were followed and testing was done after iterations. Some difficulties related to the jBPM Web Designer were found and due to the component complexity they weren't trivial to resolve.

As with most complex systems there was some trial and error during implementation and wasn't easy to efficiently connect all the different elements. Although the user doesn't realize from using the Centralized System the mxfSpeedRail products are quite distinct and have particular subtleties. A lot of hairsplitting debugging had to be done, sometimes even to the degree of comparing complex HTTP packages.

The defined schedule for each feature implementation was respected and although some took more time than planned the iteration period was always calculated considering possible delays.

The final prototype does the intended job of implementing all the required modules.

## 5. Conclusions and Future Work

Users have become extremely demanding when dealing with computer systems. They want to do more with less effort and have access wherever they are. Even if a system is very complex with a lot of technical layers and complicated features, the user won't give it a chance unless the experience has been streamlined. Furthermore, the system should also have an adaptation for mobile devices and require as less configuration as possible. Applications need to be agile and provide the user with a plug and play feeling. This notion has to be applied even in an environment as complex as TV Broadcasting. Aware of this, MOG realized that their mxfSpeedRail products required a new solution to unify the user experience.

TV Broadcaster use MOG's mxfSpeedRail products during capture and file based tasks. When the broadcaster has a small number of equipment, interacting with each one is relatively simple but when that number increases the user experience is hampered considerably. The solution to this problem was a managing system to supervise MOG's products.

By having a Centralized System, users can control and access features in a unified and simpler environment. While the number of equipment could increase the user experience stayed simple. Due to the requirements of connectivity, availability and usability of modern systems, a Web application was the best choice. The platform agnostic nature of web development allowed running the implemented solution in multiple environments.

The Centralized System exposes the mxfSpeedRail products as services. Instead of knowing what features each equipment has, the user only needs to know what action should be performed and let the Centralized System decide which product is more suited for the task. For instance, if a journalist wanted to capture an event, edit it and then broadcast the result, he only needed to design the workflow and leave the heavy lifting to the Centralized System. Knowing which equipment is able to perform the different activities, balancing the workload and assigning operations is the Centralized System's responsibility. In order to implement these complex features a Workflow Engine and a full featured business process designer were used.

The first part of this project was a deep analysis about what kind of technologies already existed and how they could be used to achieve the Centralized System solution. It also included understanding the system's requirements which helped deciding the implementation process. The second part was the development work which was made using an Iteration approach due to the application's modular nature. The iterations cycles started with an initial planning to decide which application module would be implemented followed by requirements specification and the actual implementation. At the end of each iteration, testing was done and incomplete features had to be carried over to the next iteration.

### 5.1 Goal Achievement

This thesis required a thorough analysis of SOA technologies in order to fully understand the mxfSpeedRail organization and to choose the implementation tools. As a realization of the SOA approach, a SOAP interface was implemented to interact with the mxfSpeedRail products and a business process management system was responsible for orchestrating these interactions. With this organization, the system was able to scale properly and allow complex workflow executions. All of these features and capabilities were implemented in the Centralized System prototype.

The developed prototype allows the user to perform all the planned features for the Managing System to supervise multimedia equipment:

- Has a unified access point
- Allows designing and running workflows
- Has interoperability through multiple resources types
- Balances workload during operations assignment
- Provides overview and configuration tools
- Has a global Asset Explorer

In order to implement the workflow design and execution features, a business process management system called jBPM was used together with the respective BPMN2.0 Web Designer.

To sum up, all the goals inherit to the implementation of the Managing System prototype, were completed.

## 5.2 Future Work

There is always room for improvement and this project is no exception. Due to time restrictions and priorities some features could use further developing.

The load balancing algorithm always chooses the resource with less uncompleted tasks to run the next operation. The problem with this approach is that tasks have different completion times, which means that a resource with 3 running tasks might finish faster than another with only 1 task. This might happen simply because of task complexity. To improve the work balancing feature, the algorithm should have in consideration the number of uncompleted tasks and how long each will take to complete.

Support for more complex business processes is also important. While parallel and sequential workflows work correctly, it's not possible to use human-tasks. This means that activities must be configured prior to running the workflow.

Other improvement area would be the design. Due to the framework nature of this project, the application aesthetics didn't receive much development time.

# References

- Activiti.** Activiti FAQ. *Activiti*. [Online] [Cited: November 22, 2011.] <http://www.activiti.org/faq.html#WhatIsBpm>.
- Allen, Rob.** *Workflow: An Introduction*. In: *Workflow Handbook*. s.l. : Workflow Management Coalition.
- AMQP. 2011.** amqp. *Advanced Message Queuing Protocol*. [Online] November 16, 2011. [Cited: November 17, 2011.] <http://www.amqp.org/about/what>.
- AMWA Wiki. 2011.** Main Page. *Wiki*. [Online] November 3, 2011. [Cited: November 13, 2011.] [http://wiki.amwa.tv/ebu/index.php/Main\\_Page](http://wiki.amwa.tv/ebu/index.php/Main_Page).
- AMWA-EBU. 2011.** *Framework for Interoperable Media Services FIMS Media SOA Framework 10.0*. s.l. : AMWA-EBU, 2011.
- Apache.** *Apache ODE*. [Online] <http://ode.apache.org/creating-a-process.html#UserGuide-DeployingaProcessinOde>.
- Apple.** Open Source Apple Development. [Online] Apple. [Cited: May 22, 2012.] <https://developer.apple.com/opensource/>.
- Atherton, Michael. 2002.** 2002, Darwin Magazine.
- Bouquet, Frédéric, BOS Consultant BONITA SOFTWARE FORUM.** [Online] [www.bonitasoft.org/forum](http://www.bonitasoft.org/forum).
- BPMN: An introduction to the standard.* **Michele Chinosi, Alberto Trombett.** Computer Standards & Interfaces.
- Chappell, David. 2010.** Introducing Windows Communication Foundation in .NET Framework 4. *Microsoft Developer Network*. [Online] March 2010. [Cited: January 24, 2010.] <http://msdn.microsoft.com/library/ee958158.aspx>.
- FIMS initiative.** AMWA-EBU. [Online] [Cited: June 13, 2012.] [http://wiki.amwa.tv/ebu/index.php/Main\\_Page](http://wiki.amwa.tv/ebu/index.php/Main_Page).
- Google.** ServerPushFAQ. [Online] [Cited: June 15, 2012.] <http://code.google.com/p/google-web-toolkit-incubator/wiki/ServerPushFAQ>.
- Harris Corporation. 2011.** Video Disk Control Protocol. *Wikipedia*. [Online] August 20, 2011. [Cited: November 12, 2011.] [http://www.harris.com//view\\_pressrelease.asp?act=lookup&pr\\_id=476](http://www.harris.com//view_pressrelease.asp?act=lookup&pr_id=476).
- IBM.** Introducing the Java Message Service. *developerWorks*. [Online] [Cited: January 23, 2012.] <http://www.ibm.com/developerworks/java/tutorials/j-jms/>.
- Implementation of Process Management and Control Based on JBPM4.4.* **Jian Hu, Zhiqiang Zhao, Zhongnan lv.** 2011 Second International Conference on Networking and Distributed Computing.
- Intalio.** Business Process Management. *Intalio*. [Online] [Cited: January 04, 2012.] <http://www.intalio.com/bpms>.
- International Telecommunication Union. 2011.** Abstract Syntax Notation One. *Wikipedia*. [Online] November 17, 2011. [Cited: November 17, 2011.] <http://www.itu.int/ITU-T/asn1/introduction/>.
- jBPM.** jBPM. *jBoss*. [Online]
- Liempd, Tijs Rademakers and Ron van.** *Activiti in Action*.
- Mark Richard, Richard Monson, David A. Chappell.** *Java Message Service*.
- Microsoft. 2011.** Microsoft Message Queuing. *Wikipedia*. [Online] September 21, 2011. [Cited: November 17, 2011.] <http://msdn.microsoft.com/en-us/library/ms834460.aspx>.
- MOG Solutions. 2006.** MOG Solutions. *WHITE PAPERS*. [Online] 2006. [Cited: November 16, 2011.] [http://www.mog-solutions.com/img\\_upload/PDF/MOG\\_Solutions\\_mxf\\_overview\\_by\\_mog\\_solutions.pdf](http://www.mog-solutions.com/img_upload/PDF/MOG_Solutions_mxf_overview_by_mog_solutions.pdf).

- Monteiro, Professor Miguel. 2010.** Distribution and Integration Technologies. *Distribution and Integration Technologies*. [Online] 2010. [Cited: November 16, 2011.] <http://paginas.fe.up.pt/~apm/TDIN/docs/tin12.pdf>.
- Mozilla.** Introduction to Object-Oriented JavaScript. [Online] Mozilla. [Cited: May 22, 2012.] [https://developer.mozilla.org/en/Introduction\\_to\\_Object-Oriented\\_JavaScript](https://developer.mozilla.org/en/Introduction_to_Object-Oriented_JavaScript).
- OASIS.** Web Services Business Process Execution Language. [Online]
- OMG. 2011.** Business Process Modeling Notation. *Wikipedia*. [Online] November 15, 2011. [Cited: November 20, 2011.] <http://www.omg.org/bpmn/Documents/FAQ.htm>.
- Peter, Ian. 2003.** Internet protocols history. *Internet History*. [Online] 2003. [Cited: November 13, 2011.] <http://www.nethistory.info/History%20of%20the%20Internet/protocols.html>.
- Robots Technology.** Content Agent. *Robots Technology*. [Online] [Cited: November 16, 2011.] [http://www.root6technology.com/products/ContentAgent/documents/Automating\\_file\\_based\\_workflows\\_whitepaper.pdf](http://www.root6technology.com/products/ContentAgent/documents/Automating_file_based_workflows_whitepaper.pdf).
- Salatino, Mauricio.** *jBPM Developer Guide*.
- Telestream. 2011.** Video Workflow Automation. *Telestream*. [Online] 2011. [Cited: November 13, 2011.] <http://www.telestream.net/vantage/overview.htm>.
- Tony Andrews, et al. 2011.** Business Process Execution Language. *Wiki*. [Online] November 4, 2011. [Cited: 20 November, 2011.] <http://download.boulder.ibm.com/ibmdl/pub/software/dw/specs/ws-bpel/ws-bpel.pdf>.
- VCIS.** *Improving Automation Efficiency*.
- Wadle, John.** *Workflow and System Integration in the Content Management Process*.
- WebSocket.org.** WebSocket org. [Online] [Cited: May 23, 2012.] <http://www.websocket.org/>.  
When can I use. [Online] [Cited: May 23, 2012.] <http://caniuse.com/#search=websocket>.
- Wikipedia. 2011.** Business Process. *Wikipedia*. [Online] 2011. [Cited: November 19, 2011.] [http://en.wikipedia.org/wiki/Business\\_process](http://en.wikipedia.org/wiki/Business_process).
- . **2011.** Deinterlacing. *Wikipedia*. [Online] October 22, 2011. [Cited: November 16, 2011.] <http://en.wikipedia.org/wiki/Deinterlacing>.
- . **2011.** XPDL. *Wikipedia*. [Online] 2011. [Cited: November 20, 2011.] <http://en.wikipedia.org/wiki/XPDL>.
- . **2011.** YAWL. *Wikipedia*. [Online] 2011. [Cited: November 20, 2011.] <http://en.wikipedia.org/wiki/YAWL>.