Analysis of the use of a workflow engine for OTRUM system software

by

Mohamed Altamimi

Thesis in partial fulfilment of the degree of Master in Technology in Information and Communication Technology

Agder University College
Faculty of Engineering and Science

Grimstad
Norway

May 2007

Abstract

Workflow engines are attracting more and more attention. Applications based on workflow engine technology are currently developed and deployed by many companies, such as OTRUM Company. In this project, we focus on the analysis and development of an efficient workflow engine for interactive TV.

The research project will realize workflow engine solutions based on three choices including commercial workflow engine, open source workflow engine, and a workflow engine implemented from scratch.

In this project, we go through all mentioned solutions. In the literature, many proposed workflow engines are involved for each solution. We picked one example for each choice after several tests.

For the commercial workflow engine, we adopted TIBCO BusinessWorks because of its efficiency and simplicity to use. The next step was to look for an open source solution. We suggested the JSPASM package, which is developed under Java environment. The third solution consists of developing a new workflow engine from scratch. Thus, we developed an example under the Eclipse environment to sketch some scenarios used by OTRUM.

A comparative study between the proposed solutions has been elaborated. The open source workflow engine is proved to be experimentally beneficent. It could be of interest for OTRUM.

Keywords: workflow engine, interactive TV, JSPASM, business process, state machine

Preface

This project concludes two years of master studies in Information and Communication Technology (ICT) at Agder University College (AUC), Faculty of Engineering and Science in Grimstad, Norway. This project has been carried out in OTRUM AS, from January to June 2007.

Firstly, I would like to use this opportunity to thank Prof. Andreas Prinz, my supervisor, for very good supervision and contributing opinion and advices. Secondly, I would like to thank, Per Ivar Pedersen at OTRUM Company in Arendal for good guidance and helping. Also I would like to thank, Sissel Andreassen for sympathy. Finally, I would like to thank Head of Master program, Stein Bergsmark for helpful feedback.

Table of Contents

List of figures

List of Tables

List of charts

Chapter 1 Introduction

Many solutions and proposals exhibit a good performance. An excellent workflow engines exist today [1-7]. They differ from one another because they are generally tailored toward different application domains and different user requirements. While the engine authors have often painstakingly met a specific set of requirements driven by a particular application, and thus have provided something of deep value to a particular user community, these systems can be difficult to adapt to a new domain.

In the last decades, the migrations towards applications which are fully based on workflow engines have increased. Scientists and researchers are putting emphasis into this area. However, an excellent and complete solution has not been achieved yet. Many existing solutions and proposals exhibit good performance today, some of which demonstrate excellent workflow engines [1-7]. They differ from one another because they are generally tailored toward different application domains and different user requirements. While the engine authors have often painstakingly met specific requirements set by a particular application, providing something of deep value to a particular user community, the systems can be difficult to adapt to a new domain.

Arguably, the commercial community is more advanced than science community in defining and standardizing workflow languages that at least hold the potential of interoperability. Typical workflow scenarios include document lifecycle management, internal application workflow and business process management.

Nowadays, workflow engine approach tends to be more adaptable and convenient for different applications. The principle of a workflow engine is inspired by the state machine concept where many processes can interact together in order to accomplish a function. The transition between states is even more important for some applications. In addition, the process can switch from a state to another. On the contrary, the classical programming way holds on the sequential principal.

The workflow engine was investigated theoretically and experimentally. Therefore, many programming languages like for instance Java, Delphi, etc. are currently incorporating new tools for workflow implementation with a user-friendly interface. JWT is an example of a workflow engine, which will be ready to use in the foreseeable future.

Due to this interest, many companies have invested in workflow engines, so commercial workflow engines can be found in the market. Nonetheless, some open source workflow engines exist as well.

This chapter will introduce some of the terms used in the thesis. We will first describe OTRUM, where we carry out this current project. Afterwards, we will shortly describe the OTRUM system. Finally, the problem description will be stated, as well as the delimitations that are supposed to be taken in consideration.

1.1 The OTRUM Company

This project is given by OTRUM, situated in Arendal. OTRUM is one of a few companies rapidly expanding all over the world. OTRUM is a leading provider of interactive TV solutions and content for hotels and hospitals. The company develops and supplies total solutions, consisting of hardware, software, TV channels, films and Internet access. [8]

The company was established in 1985 by two Norwegian entrepreneurs who saw the possibility of in-room interactive television for hotels. Scandinavia was the first target market, but OTRUM soon attracted the attention of European hotel chains. Today, over 97% of sales are to export markets. OTRUM is now a multinational company, which has a distribution network and many subsidiary sales that cover Europe, the Middle East and Africa.

OTRUM has partnerships with leading TV vendors, to ensure full integration of in-room services. In 2001, OTRUM created its own direct distribution network with Telenor, its new major shareholder. This was a new era for OTRUM in that time. The figure below gives an overview of the link-to-home solution, providing interactive TV systems, TV channels, movies and others services.

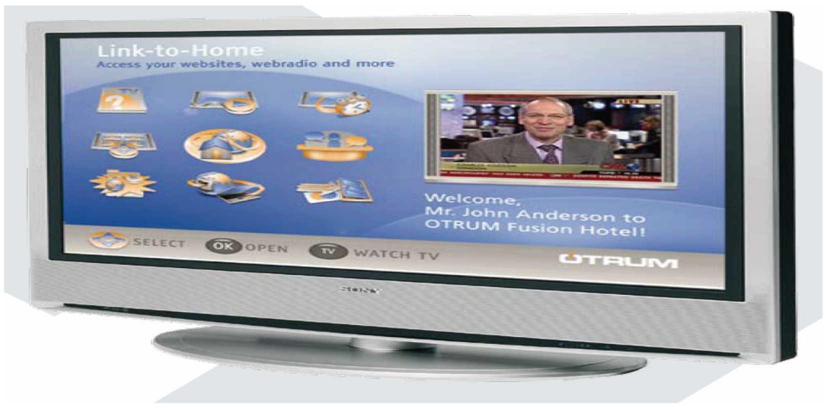

Figure 1-1: New level of interactive TV entrainments

1.2 OTRUM `s System

OTRUM Fusion is the latest interactive product form OTRUM, built upon their architecture of existing OTRUM interactive platform. The new level of interactive TV entertainment encompasses the latest streaming and Internet technologies offered by OTRUM Fusion.

The system software of OTRUM relies on a code that dates to 1992. Since that date, gradual changes in the system has occurred. In this perspective, new services were added, more convivial graphical interfaces were designed, and new TVs were integrated. Most recently, support for OTRUM`s Fusion Digital Client has been introduced. In addition to the development environment of OTRUM is based on Delphi. Delphi is object oriented programming language propriety of Borland that was derived from Pascal. It makes OTRUM less attractive and the code structure of OTRUM is extremely complex. Thereby, the maintenance and the modification of the code is a highly complicated task that threatens to overwhelm the programmer attention. Moreover, the code does not support testability. In fact, testing is time consuming and cannot handle all the system configurations. Furthermore, automatic testing cannot be performed on the system while running. Many service configurations are never used adding unnecessary complexity to the system.

From this perspective, a main challenge that rises is fuelling the efficiency of OTRUM. Therefore, building a new platform based on the current system should be carefully investigated. This new platform is intended to simplify the system architecture and should be oriented towards openness by integrating standard protocols and tools. A focal concern that should be considered in this platform will be a workflow engine.

1.3 Master task

Current OTRUM system software is based on a code that was first developed in 1992. It has evolved by porting to Windows, addition of new services, transition to a graphical interface, support for new TVs and at last support for Fusion digital clients.

The language with associated tools used in OTRUM development environment now is Delphi from Borland, an object-oriented extension of Pascal. The tools are not widely used in the industry, leading to lower availability of third party libraries and making OTRUM less attractive when recruiting personal.

The long history of the system software has led to several problems with maintenance and testing of configurations. In addition, it is difficult to find developers willing and able to handle this code.

Out of these problems OTRUM has started to design new system software. Hereby the target is to simplify the system and to use more standard protocols and tools. A major component in a new platform will be a workflow engine.

This task is about the selection of an appropriate workflow engine for the new OTRUM system software which will be developed in a Java environment. The selection has to be based on requirements provided by and discussed with OTRUM. Central questions are:

- Is it possible to use a 3rd party workflow engine stage? Which one?

- Is it best to invent an own workflow engine from scratch? If yes, what will a suitable architecture be?

- How can we integrate the workflow engine with the rest of the system software?

1.4 Problem description

In order to adapt OTRUM system to fast technology changes, OTRUM has decided to perform updates on their system in such a way that it becomes easily updatable. The current OTRUM system has several drawbacks including:

- Complexity: Currently OTRUM's system is very complex, making it difficult to maintain and test a given configuration.
- Modernity: OTRUM's development environment uses the Delphi language, and tools that are not widely used in the industry.
- Developers: It is difficult to find developers willing and able to handle this code.
- Attractiveness: Delphi makes OTRUM less attractive because of lower availability of third party libraries.

1.5 Delimitations

During the discussion with OTRUM and according to the requirements related to the core task of this thesis, some fundamental assumptions have a limiting impact on pursuing our research task. Here, we list these constraints.

- Use of the existing OTRUM Fusion interactive platform
- Operative system has to be Linux
- Development language is Java
- UDP should be the media protocol

1.6 Purpose

The aim of the project is the selection of an appropriate workflow engine for the new OTRUM system that overcomes the shortcomings and errors of the current system. The new OTRUM system software will be developed under the Java environment. It should guarantee higher performance and use more standard protocols and tools.

1.7 Audience

The main audience of this project is OTRUM, in addition to company members and others that are interested in workflow engines and the technical aspect and motivations behind it. This document is intended to be easily understood by a moderate technical audience without the need of an intensive prior knowledge of workflow systems.

1.8 Thesis Overview

This thesis is structured as followed. After introducing general information about our current project, chapter 2 depicts the history of the OTRUM system and its main features as well as the background required in this project. In chapter 3, we put emphasis onto the state-of-art. The existing solutions for a workflow engine are listed altogether with their characteristics. Chapter 4 deals with the implementation of our proposed workflow engine based on the JSPASM package for Java eclipse. There, we sketch some useful scenarios for an interactive TV recommended by OTRUM. After that, in chapter 5 we show the related results issued from our implementation. To proceed, we create a simple user-friendly interface and a workflow engine that manages all presented services (wake up, movie-on-demand…). In chapter 6, we discuss the obtained results and to which extent the project goals are achieved. Finally, we conclude our project by noting the highlights of our work and considering some horizons and perspectives for a future work.

Chapter 2 Background

Recently, many companies have begun moving towards systems that are fully based on workflow engine technologies. OTRUM aims to deploy such technologies. Previously, OTRUM has developed an interactive TV based on classical programming (sequential programming). Actually, due to the drawbacks of the old system, OTRUM engineers are trying to substitute the old system with a new system that is 100% based on workflow engines. So far in this section, we will present in the first instance the old system that is currently functional. In the second instance, we will demonstrate briefly the different features of the new expected system.

2.1 Old system

The old system is principally developed under the Pascal environment. It is comprised of two parts: a hardware part, which contains information about devices and adaptors, and a software part that drives all functionalities that aim to make the whole system work properly. The figure below shows an overview of the existing system.

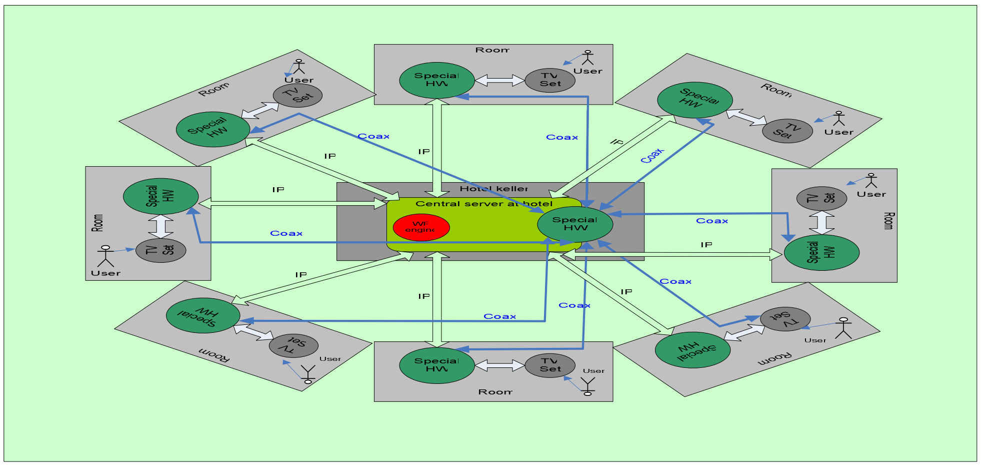

Figure 2-1: OTRUM old system

The example depicted in figure 2-1, presents an instance of OTRUM architecture. It involves some components that are listed in following sections.

2.1.1 Central server

OTRUM system involves an important component that plays the role of orchestra leader in the whole system. It is called the central server. It is located in the centre of each building. It has the following functions:

- It has to acquire requests from the TV Room for a service
- It handles video streaming when the customer demands movies service
- It holds supervising and maintenance functions in case of an outage.
- It makes updates to the whole system.

2.1.2 Special hardware

The specific hardware is a special device developed by OTRUM in order to ensure the communication between TVs and the central server. The main functions of this device are:

- Electrical functions (power adaptation, modulation, coding...)
- Uses Coax cable to attach TVs with the central server
- Encryption
- Link level functions (error handling, flux control)

2.1.3 Delphi

As noticed before, the OTRUM system was developed under the Pascal environment. It makes use of Object Pascal. According to literature found, Object Pascal is an object oriented programming language derived from Pascal. Object Pascal is often referred as the father of Borland Delphi.

The first versions of Borland Delphi were called Object Pascal; before that, Borland decided to replace this name with “Delphi programming language”. Due to the close relationship between Object Pascal and Delphi, most of the dedicated compilers to Object Pascal support Delphi source as well.

Today, integrated development environments (IDEs) for Delphi are available on most platforms, including Microsoft Windows, the Microsoft .NET Framework and Linux.

Therefore, the source code of Delphi can be compiled for Linux, Mac OSX, Win64, Windows CE, and others. [9]

2.2 Expected new system

Since the old system did not respond to the user needs and did not follow the revolution, OTRUM decided to find an efficient solution to overcome the drawbacks of the old system. Hence, they built up a new prototype for a new system that supposed to be more reliable and scalable.

OTRUM profited from workflow engine technologies by including workflow engine features on its system. Moreover, they implemented new protocols like TCP/IP protocol. They migrated also towards Java environment so that the system could be more interoperable and open. The following figure depicts the architecture of the new solution for OTRUM interactive TV system.

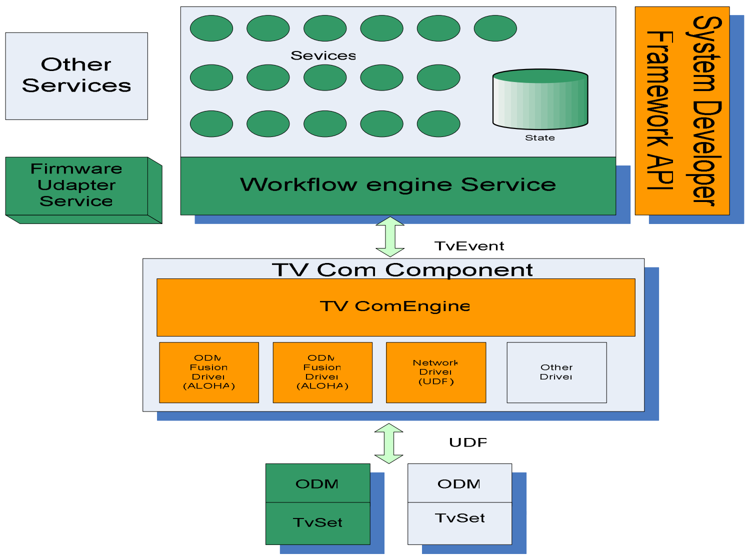

Figure 2-2: The new expected system

2.2.1 Workflow engine

As it is named, workflow engine is a compound name consisting of: workflow and engine. "Workflow is concerned with the automation of procedures where documents, information or tasks are passed between participants according to a defined set of rules to achieve, or contribute to, an overall business goal [1-7][10].” An overview of the mechanism of a workflow system according to the definition is shown in the figure below.

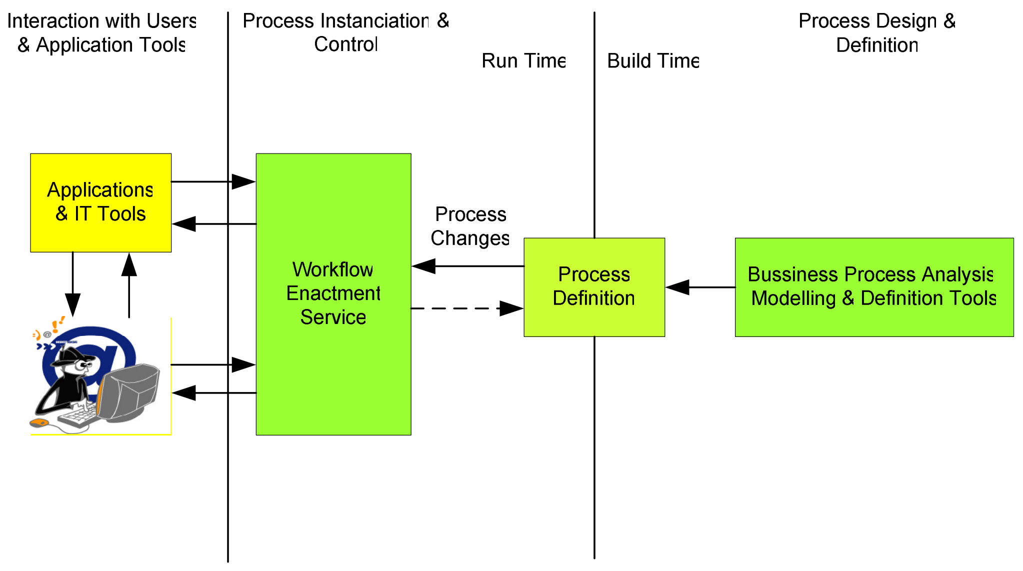
Interaction with Users & Application Tools
Process Instanciation & Control
Run Time
Build Time
Process Design & Definition
Applications & IT Tools
Workflow Enactment Service
Process Changes
Process Definition
Bussiness Process Analysis Modelling & Definition Tools

Figure 2-3: workflow system characteristics

Workflow is often associated with Business Process Re-engineering, which is concerned with the assessment, analysis, modelling, definition and subsequent operational implementation of the core business processes of an organisation (or other business entity).

Workflow engine is a software application meant to manage and execute modeled business processes. In the following, we list the main features of a workflow engine.

- ✓ “Control of process instances- creation, activation, suspension, termination etc…
- ✓ Navigation between process activities, which may involve sequential or parallel operations, deadline scheduling, interpretation of workflow relevant data.
- ✓ Access management of specific participants
- ✓ Maintenance of workflow control data and workflow relevant data, passing workflow relevant data to/from applications or users.
- ✓ An interface to invoke external applications and link any workflow relevant data
- ✓ Supervisory actions for control, administration and audit purposes. [5]”

A prototype of a model workflow reference involving the workflow engine is presented in the figure 2-4. It comprises different components that interact together so that they accomplish a complete workflow system.

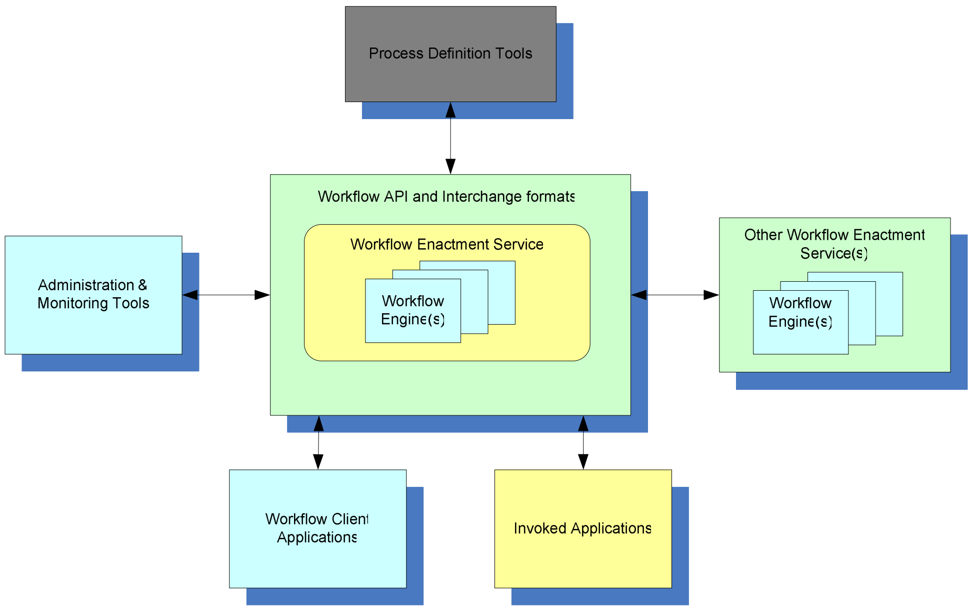
Process Definition Tools
Workflow API and Interchange formats
Workflow Enactment Service
Workflow Engine(s)
Administration & Monitoring Tools
Other Workflow Enactment Service(s)
Workflow Engine(s)
Workflow Client Applications
Invoked Applications

Figure 2-4: Components for a workflow model

2.2.2 Business process

The business process is a part of a workflow system. It is defined as a set of related activities that assign an output value for each input. Both input and output can be formed as object and the transformation action is performed by human actors, machines or both. In the literature, we found three possible kinds of business processes:

- ✓ Management processes
- ✓ Operational processes
- ✓ Supporting processes

In the following, we notice the relationship that exists between our subject and a business process. Here, we sketch the scenario of the whole interactive TV system with consideration for the business process concept. It is pointed out that our architecture is mainly centralized. Therefore, possible business processes could be added to the server as an extra function in order to manage several services, each related to a business process.

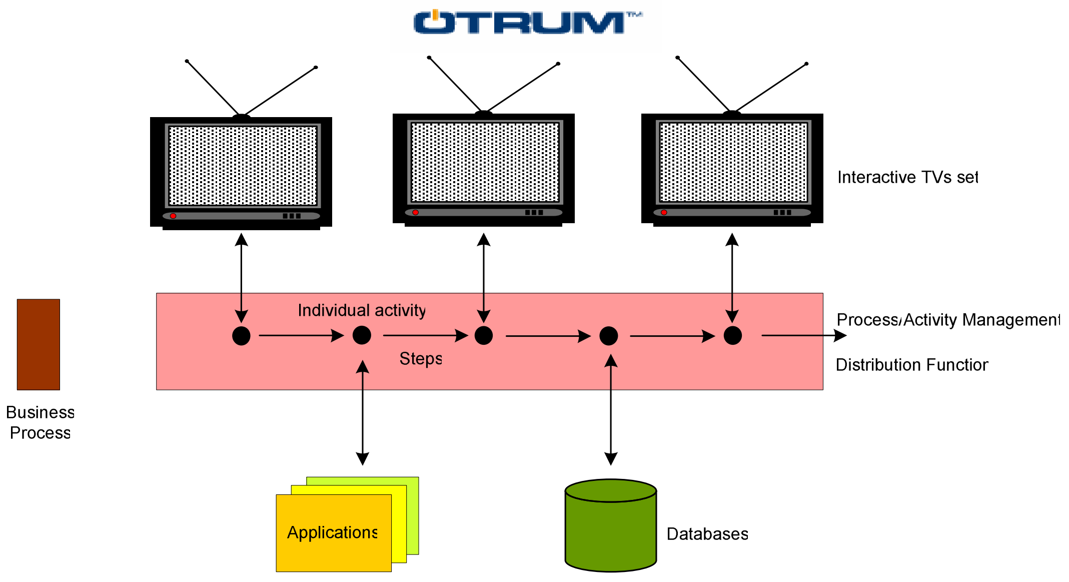

Figure 2-5: Example of deployment of business process

2.2.3 ODM

The ODM is an electronic card that plays the role of an interface between the TV set and its com component, these two devices exchange information using the UDP protocol. The TV Com Component consists of a TV COM Engine and different drivers. The ODM behaves like a de-multiplexer, directing the information to a given driver based on the action needed to be performed. In addition, the ODM is responsible for displaying the stream on the TV set.

2.2.4 ODM Fusion Driver (ALOHA)

The ODM fusion driver receives packets from the ODM device, encapsulates, and forwards them to the TV COM Engine in a format that is understandable for this device.

2.2.5 UDP (User Datagram Protocol)

UDP (User Datagram Protocol) is a communications protocol that offers a limited amount of service when messages are exchanged between computers in a network that uses the Internet Protocol (IP). UDP is also called unreliable Datagram Protocol because it does not provide the reliability and the ordering that TCP protocol offer. Many multimedia programs like OTRUM use UPD protocol because of its low latency; UDP is faster and efficient for many lightweight or time-sensitive purposes. When compared with TCP, data arrive out of order, is duplicated, or lost without warning [11].

2.2.6 TV COM Engine

The TV com Engine is a service engine used as an interface to build up the relations between different terminals (TV Sets) and service servers. As an engine it can search different proprieties of terminals and also the services supplied by servers. As an interface, it can supply several functions such as building links between user terminals and serves core challenge for this TV Com engine. In addition, it identifies how many devices are connected to this engine and by which channels the services can be sent from (to) a starting point (a destination). The TV com Engine is considered as the most important part in the link layer.

2.2.7 TV Event

The TV event includes user actions and TV device response. In general, the TV event is trigged by a user action. In fact, the remote control can be used to search and get the services that a user wants. When the event starts, the TV set catches the wanted service and sends a service request to the server. Meanwhile, the TV set waits for the response. When the response comes from the server, the TV set supplies it to the user by showing it up on the monitor.

2.2.8 System Developer framework API

A framework is a defined support structure in which another software project can be organized and developed. A framework can include support programs, code libraries, a scripting language, or other software to help developing and connecting the different components of a software project. Frameworks allow designers and programmers to save more time on meeting software requirements rather than dealing with the more tedious low-level details of providing a working system; this is because frameworks are designed with the intent of facilitating software development. [12]

2.2.9 State

A workflow application program generates and updates data on which process navigation decisions and other operations are preformed. The related data is stored in the state.
The only data to which the workflow engine has access is the data stored in the state. This data is known as "case data".
When an application is invoked, the latter is solely responsible for the manipulation of the workflow application data. However, many applications can be invoked at the same time at different activity points within the workflow process. In this case the workflow engine is responsible for transferring data between such applications.

2.2.10 Java

Java is a programming language developed by Sun Microsystems. Java was first called OAK. OAK was a failure for the Sun. Subsequently the firm changed its name to Java in 1995. Improvements and modification were introduced making Java a powerful language compatible with the World Wide Web.

Like C++, Java is an object oriented object language. However, Java is simpler than C++. Its easy and clear syntax significantly alleviates the number of common programming errors made by developers. The Java source files are compiled into byte code that can be executed by a Java interpreter. The main characteristic feature of Java is its portability. In fact, the compiled Java code can run on most on the operating systems. This portability is ensured by the so-called Java Virtual Machine. This machine translates the code to known instructions interpretable by the underlying operating system. A keynote to underline is that a Java virtual machine comprises of a Java interpreter and a runtime environment, Java is a programming language that can be used for a wide area of applications. Java also seems very convenient when used in the World Wide Web due to a number of specific features [13].

2.3 Comparison between old and new system

The old and the new OTRUM system shares some features and differ in other features. According to figures 2-1 and 2-2, the architecture of the new system has been totally changed and it looks more flexible. The table below contains a comparative study between the old and new system.

	Common components	Particular components	Dimension	Robustness
Old system	ODM, ODM fusion driver, Medium (coax)	Transmission protocol (sockets)	Non scalable	Non flexible and complicated
New system	ODM, ODM fusion driver, Medium (coax)	Transmission protocol (TCP/IP)	Scalable	Flexible and more simpler

Table 2-1: comparison between old system and new system

Chapter 3 State-Of-Art

Today, many organizations perceive that workflow engines are inevitably needed due to their competitiveness in several areas, and there is a continual need to improve productivity. Workflow engines have been an active research and development area with several research prototypes and commercial products in the market.

In this section, we state some examples of solutions for workflow engines founded in literature. Some of these structures are more useful than others. In the state-of-art, we attempt to find out where the research is and how far we can achieve our goals. Here, we show a comparative study of existing and future workflow engines issued on the market as commercial products or as free tools.

3.1 Products selection

There are many workflow engines on the market today. These engines are so different from each other that they might not fit our requirements. In order to find the workflow engines with the requested properties, we used the Internet to look for all workflow engine producers. Here we will list the workflow engines that can or cannot achieve our goals.

3.1.1 List of commercial workflow engines

Here we will list some of commercial workflow engines:

- iMarkup [14]
- Microsoft's BizTalk Server [15]
- Microsoft's windows Workflow Foundation [16]
- TIBCO BusinessWorks [17]
- Oracle's Business Process Manager [18]

3.1.2 List of open source workflow engines

There are many open source workflow engines; each generally dedicated for single application domain. Here we list some of them:

- Spring [19]
- JSPASM [20]
- Twister [21]
- JBPM [22]
- WfMOpen [23]

More open source workflow engines could be found in [24]. To select an appropriate workflow engine among those listed above, we made a comparison between each of them. The table below shows a comparative study with regards to simplicity, interoperability, scalability and adaptability.

	Scalability	Adaptability	Interoperability	Simplicity
iMarkup	-	-	-	++
Microsoft's BizTalk Server	+	-	-	+
Microsoft's windows Workflow Foundation	++	+	-	+
TIBCO BusinessWorks	++	+	+	+
Oracle's Business Process Manager	-	+	+	-

Table 3-1: Comparison between commercial workflow engine

	Scalability	Adaptability	Interoperability	Simplicity
Spring	+	-	+	-
JSPASM	+	+	+	+
Twister	-	+	+	-
JBPM	-	-	+	+
WfMOpen	-	+	-	+
JWT	++	+	+	++

Table 3-2: Comparison between open source workflow engine

After mentioning the drawbacks and the advantages of these commercial workflow engines, we decide to pick TIBCO BusinessWorks as a testing example to implement scenarios for an interactive TV application. The same was done for the commercial workflow engines; we chose one example, the JSPASM Java package, to perform our scenarios. Later on, we will note the obtained results using different chosen solutions.

3.2 Open source workflow engines

3.2.1 WFMOpen

WfMOpen is a J2EE based implementation of a workflow facility (workflow engine) as proposed by the Workflow Management Collation (WfMC) and the Object Management Group (OMG). The component of the workflow engine is based on:

- Java interfaces that define API
- Omgcore interfaces
- Workflow are specified using WfMC`s XML Process Definition Language (XPDL)

The figure 3-1 illustrates the principle of WfMOpen structure

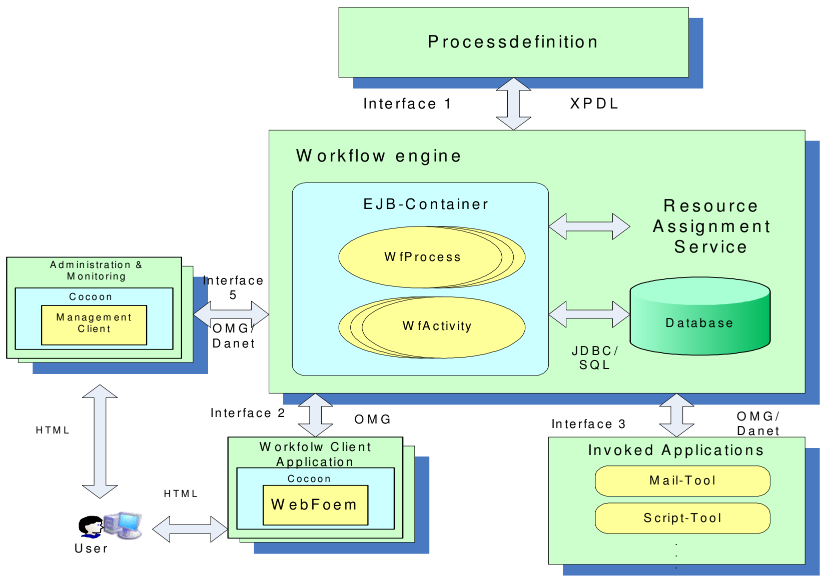

Figure 3-1: WfMOpen structure

3.2.2 Java Workflow Toolbox (JWT)

As an example of a workflow engine under an open source environment, JWT is the new upcoming workflow engine which is a plug-in for Java Eclipse. It has the following characteristics:

Workflow Editor (WE)

- Graphical representation of process definition
- Workflow engine Administration and Monitoring (WAM)
- Process Definition Management (Repository & Package)
- Process Execution Management (Instantiation, Monitoring…)
- Process Runtime Resources Mapping (Application & Users)

Figure 3-2 resumes the structure of Java Workflow Toolbox [25]

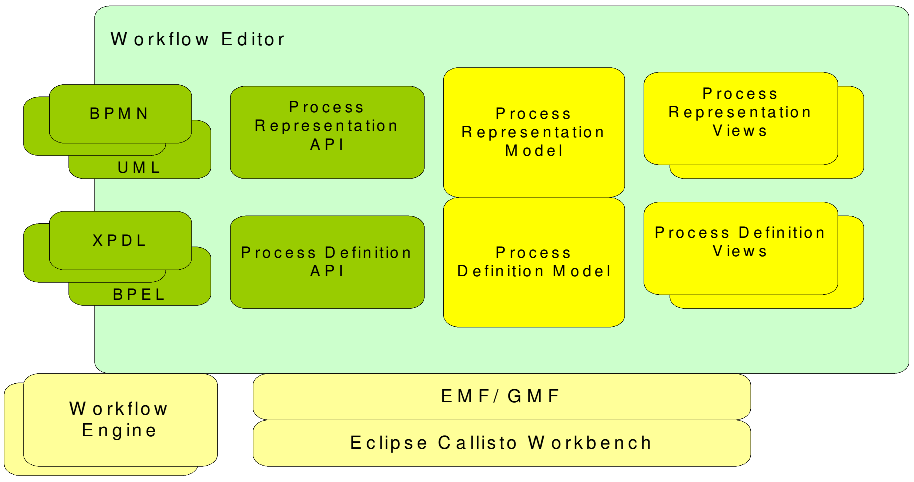

Figure 3-2: JWT structure

3.2.3 JSPASM

JSpasm (Scalable High-Performance State Machine) is a 100% pure Java™ package that makes it easy to develop applications, libraries or other packages based on state machines.

JSpasm allows the development of systems that contain multiple state models and large numbers of entities working within those state models in parallel. The java.nio package eases this problem somewhat by providing non-blocking I/O operations that allow a single thread to handle multiple sockets. However, it is left to application developers to divide a framework for handling large numbers of sockets with a limited number of threads.

JSpasm provides a java.nio package that provides non-blocking I/O operations that allow a single thread to handle multiple sockets. By associating a state model entity with each connection it becomes simple to handle large numbers of connections without requiring a thread for each [9].”

Our proposition within the current project is to use JSPASM package to develop an example of deployment of a workflow engine for interactive TV. It is difficult to compete with the commercial solutions that are definitely optimized and efficient. Therefore, according to the sate-of-art, we will exhibit in the next sections an example of implementations of Workflow engine based on JSpasm.

When writing about JSPASM, one cannot ignore writing about two important definitions, which are event-driven programs and State machine.

3.2.3.1 Event-driven programs

What is an event- driven program? The definition of event-driven is the “one in which the actions of the program are determined by events delivered to the program indicating that certain conditions have arisen” [26]

3.2.3.2 State machines

”A state machine is a system in which an entity can be in one of a finite (and usually quite small) number of states, and changes to another state (a transition) when an event is delivered to it. The state it changes to depends upon the starting state and the type of event that is delivered. Usually the state machine executes some code associated with the new state when a transition happens.”[26].

3.3 Commercial workflow engines

3.3.1 TIBCO Business Works

TIBCO BussinessWorks is ranked as an efficient solution for deploying an application with a huge amount of information. This solution performs the following features:

- Graphical interface enables users to define metadata and application services,
- Transformation rules, and process flows.
- Tester for debugging before deployment.
- Security capabilities including basic identity authentication or SSL with certificate management.
- Robust exception handling and error reporting throughout design, testing and deployment.

The figure describes shortly the principle of TIBCO Business Works.

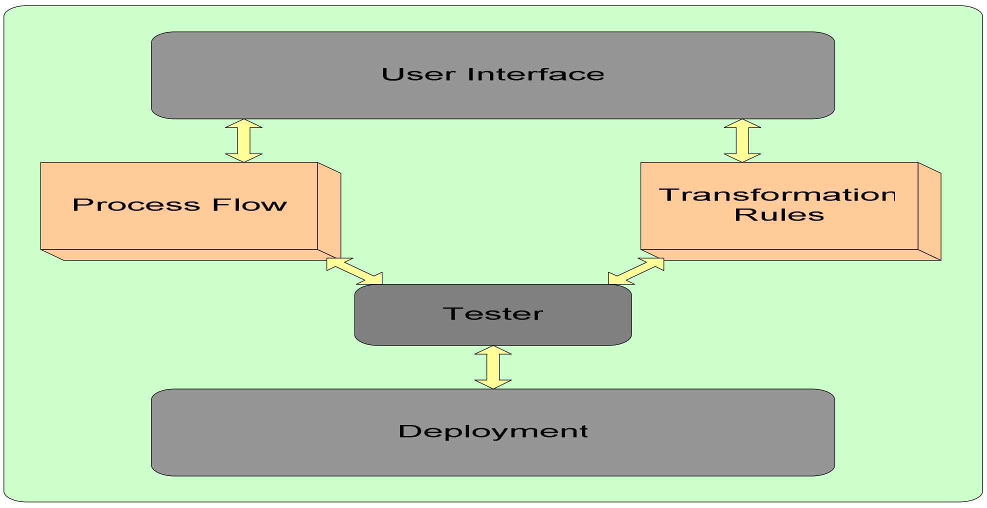

Figure 3-3: TIBCO Business Works Structure

Chapter 4 Implementation

To ensure the efficiency of our chosen workflow engine tool, an implementation of a concrete example could be of interest. In this chapter, we depict the implementation of some interactive TV scenarios such as wakeup service. Moreover, we will show their implementation under different workflow engines picked in the previous section.

4.1 Interactive TV scenarios

When using interactive TV, clients have interests in some common used services that should be present on behalf of any hotel. In this section, we will basically present some simple example related to interactive TV to give readers a better understanding of the main the functions of the system.

4.1.1 Start scenario

Generally, the first scenario that can happen, is the one regarding switching between on and off states. The first time the user turns on the TV after ordering a room, a welcome page, which is described in the next scenario, appears.
Supposing that the user turns off the TV and get in for the second time, the welcome page should not appear again. Now, the main page should show up instead. However, after leaving the room, the administrator will delete the old user name from the system's database and perform a check whenever a new user comes up.
The following figures present both scenarios described above. Figure 4-1 is devoted to the first time the user turns on the TV. The second figure 4-2, sketches a scenario where a repetition occurs.

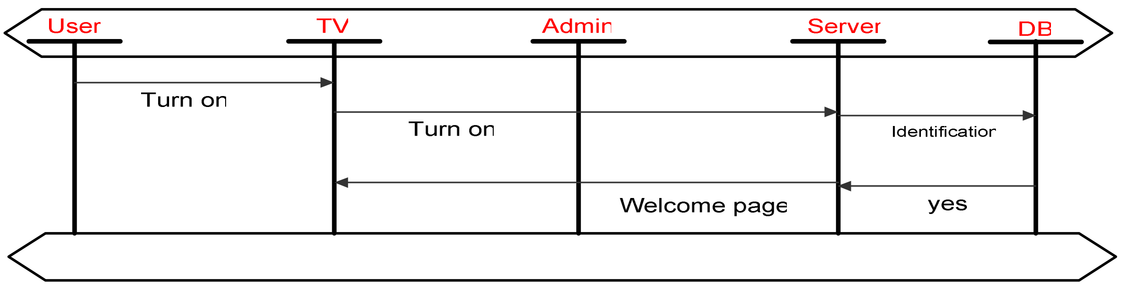

Figure 4-1: The first time TV turning on

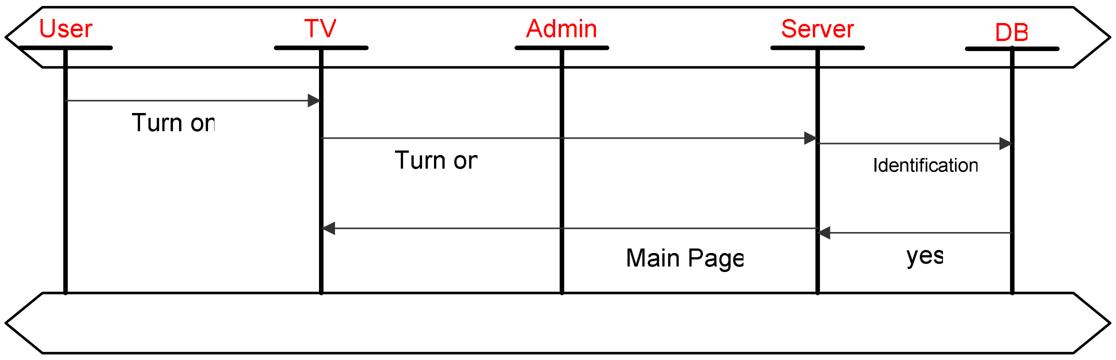

Figure 4-2: Turn on TV repetition

4.1.2 Greeting scenario

Once the user turns on the TV, a message pops up with a greeting, i.e." Welcome Mohamed". The user then clicks an OK button to leave the welcome page and proceed to the main page. The present scenario is depicted here in the form of sequential diagram. However, in the case where a new client is started, an update on the system must occur. Therefore, we should ensure the ability to register personal information related to the owner of the system. An administrator holds the updates on the server.

4.1.3 Wake-up Scenario

After finishing the scenario we presented before, the user can do whatever he wants. When a guest wants to set time for example, he first clicks on button 2, which refers to a wake up button on remote control, which leads to a wake up page. Here the user leaves the main page, proceeding to the wake up state. Figure 4-3 presents such scenario.

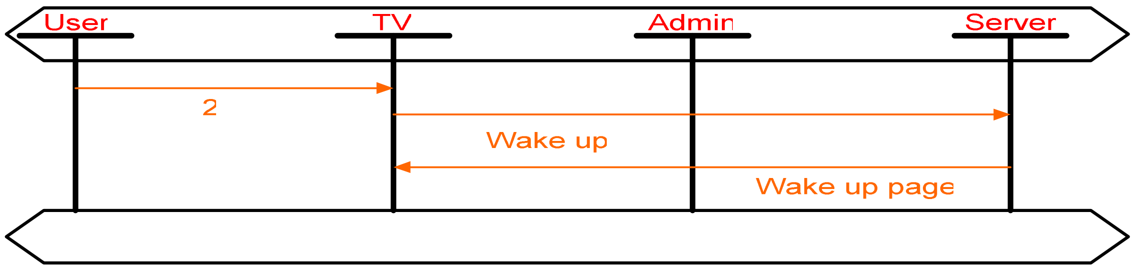

Figure 4-3: wake up page

To set the time for the wakeup service, the user utilizes the remote control. For example, to set 8 am, the user must push the buttons 0800 successively. Figure 4-6 presents the set time process.

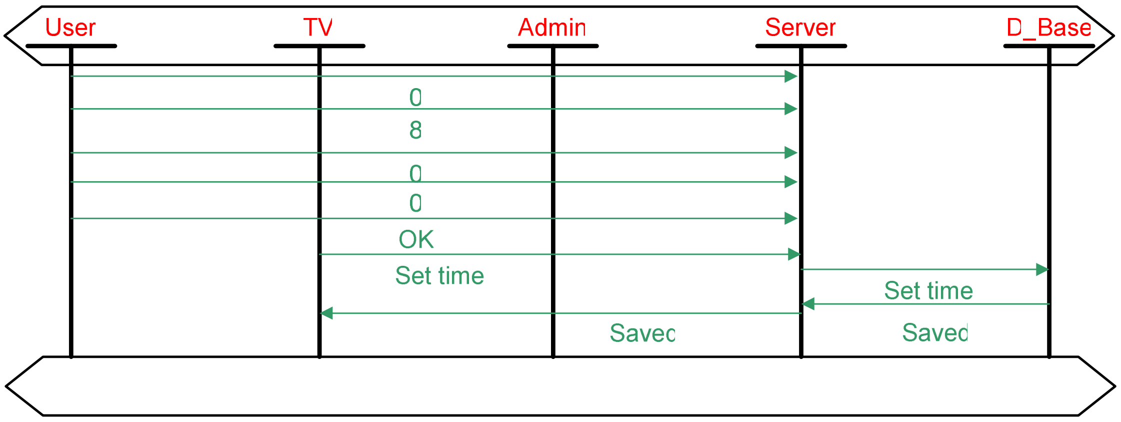

Figure 4-4: Set time process

When the user has finished setting the time, the guest can set a new wake up time by going through the same process. In our scenario, we suggest that the old wake up time will be deleted and the new time will take its place. Figure 4-5 presents one such scenario of ordering a new wake up time.

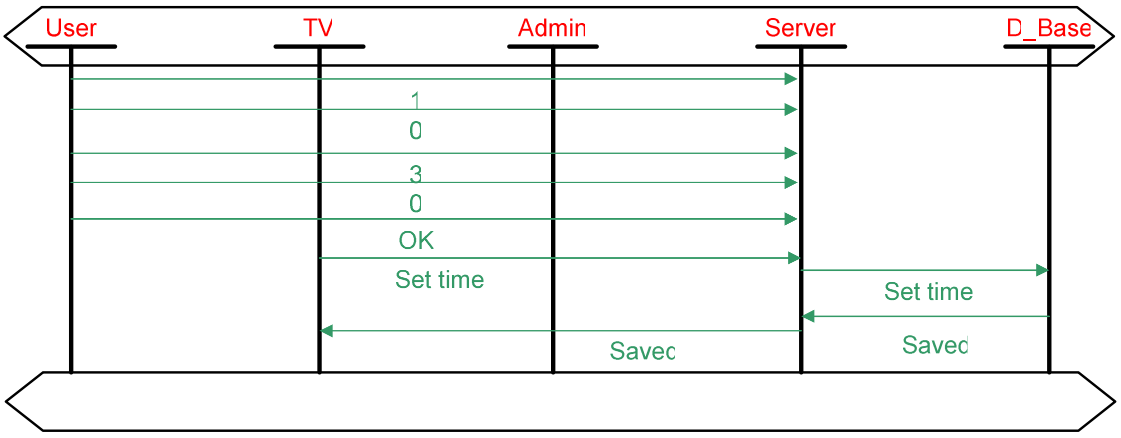

Figure 4-5: Ordering a new wake up time

If the guest types an incorrect time format (i.e. 5060), he will get an error message as shown in the following figure 4-8.

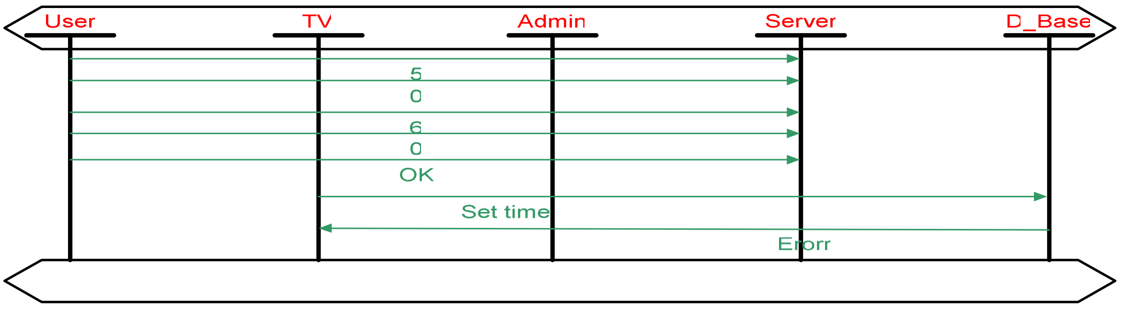

Figure 4-6: Error time format

4.1.4 Movies Scenario

In the case where the guest wants to watch a film, clicking on button number 1 on the remote control will bring up a list of films. The guest will then be able to select and choose a film. We always want to keep main page available on the screen. Figure 4-7 presents such scenario.

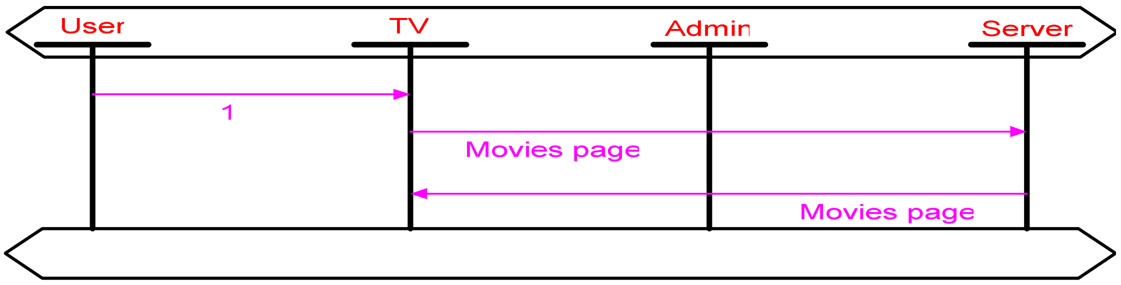

Figure 4-7: Movies page process

After the guest has entered the movie page, movies can be chosen. For instance, to pick movie number 1, the user clicks on button 1. Figure 4-10 shows an example of selecting a movie.

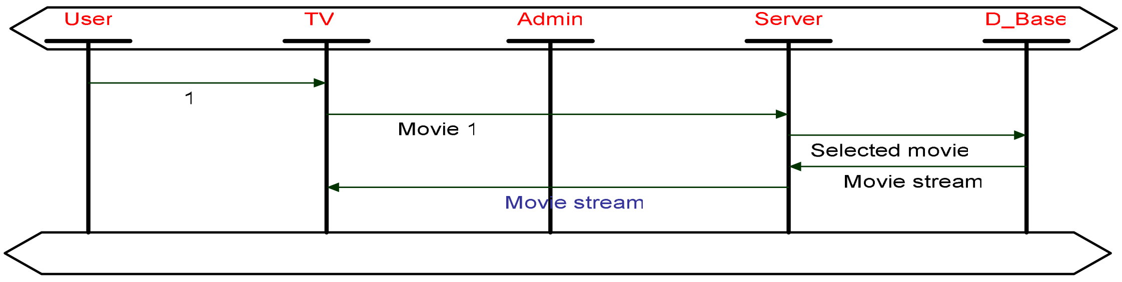

Figure 4-8: Selecting a movie

In this scenario, we expect error handling to occur. For instance, if the selected film is not available at the moment, an error message will be sent back on the user's screen.

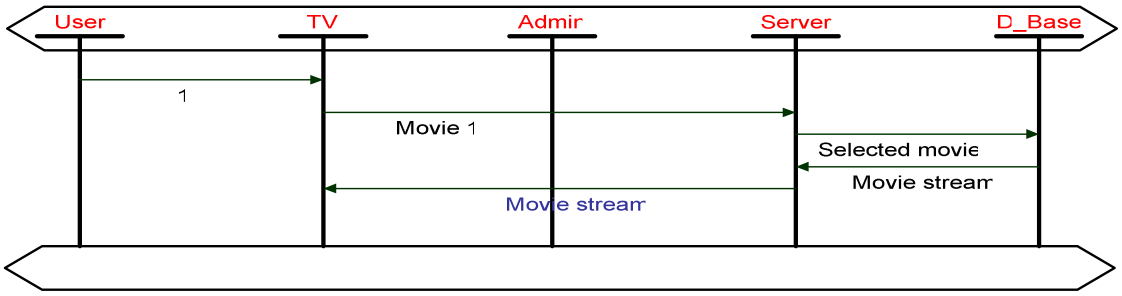

Figure 4-9: Availability of films

4.2 Other representation structure

The listed scenarios could be represented in other way as shown in the following diagram.

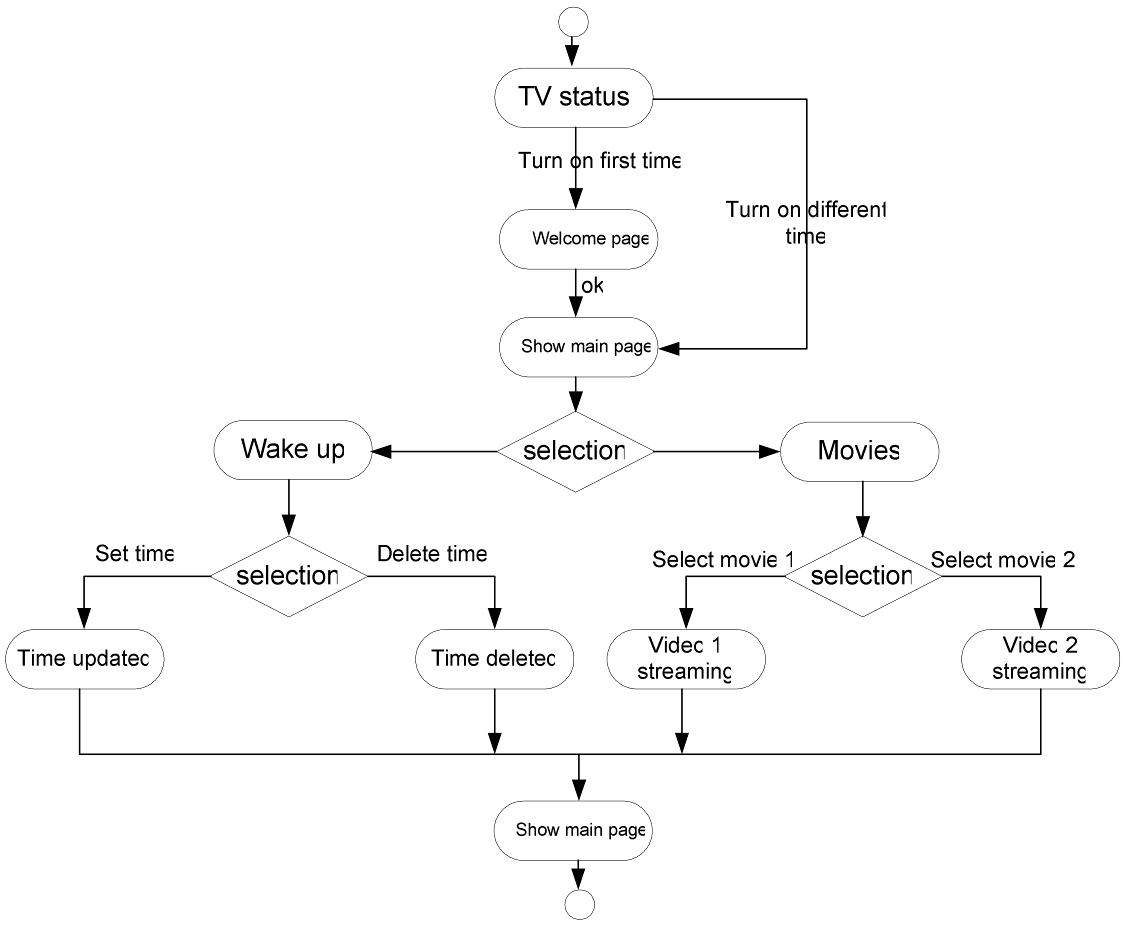

Figure 4-10: Another way to present our scenario

4.3 Testing scenarios under SDL

To prove the consistency of the interactive TV scenarios defined above, we tested them under Cinderella SDL. “Cinderella SDL is a visual modelling tool for developing embedded software, protocols and any of massage or signal based system”. In the following diagram, we show the implementation of the interactive TV system processes and different interactions that can occur.

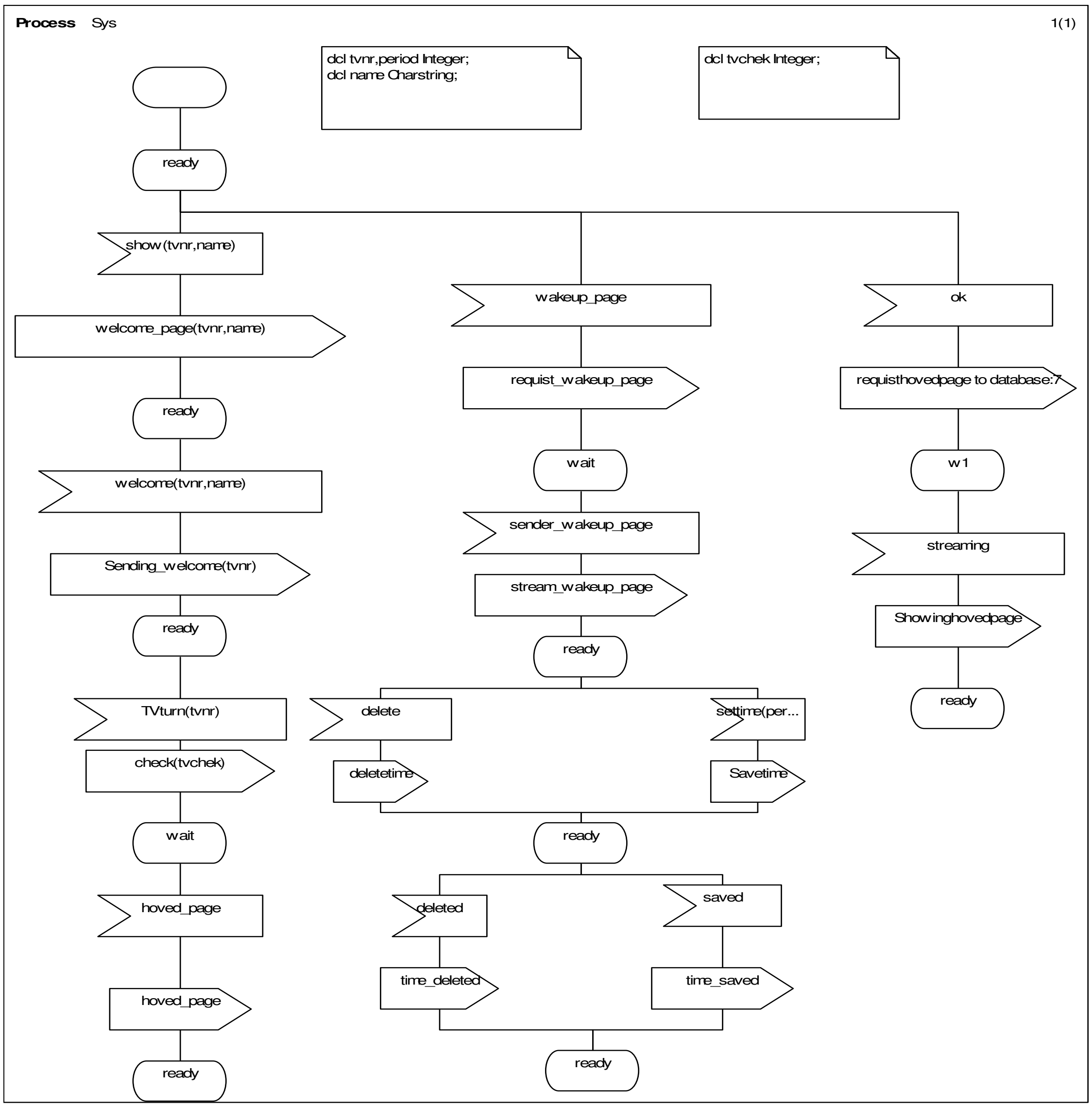

Figure 4-11: System SDL Process

4.4 Implementation using TIBCO BusinessWorks

TIBCO BusinessWorks provides a powerful user interface that lets the user manage and draw the process respecting the workflow representation. It also supplies the possibility to interact with other systems and environments such as Java. Figure 4-12 shows a screenshot of the sketching of these defined scenarios. It shows the interaction and transition between each process.

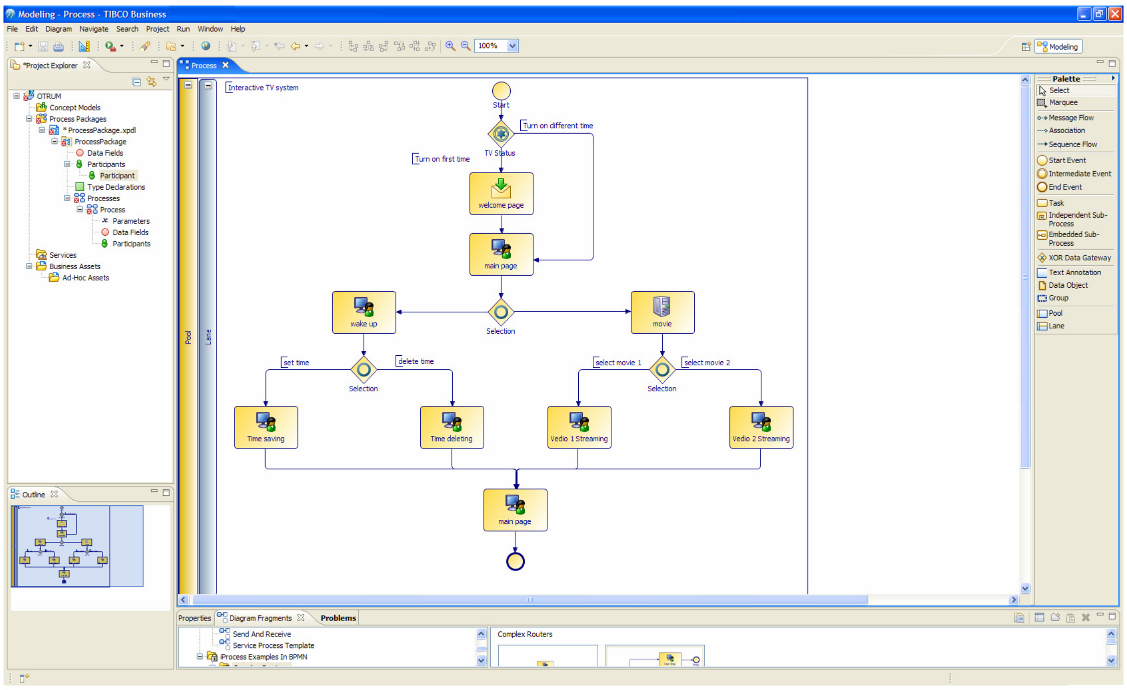

Figure 4-12: Screenshot of the implementation under TIBCO

We developed a simple user interface in Java eclipse to let user get access to the application. The screenshot in figure 4-13 shows the interface.

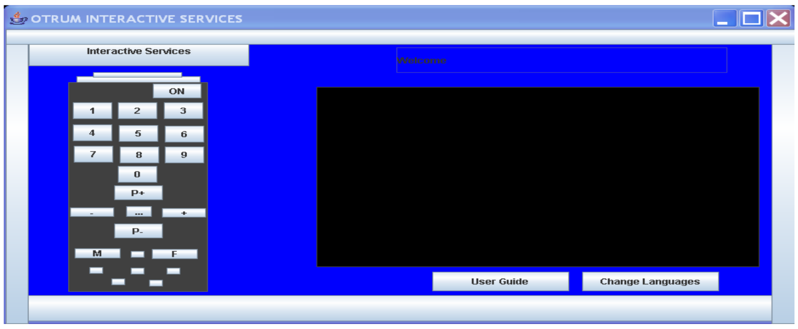

Figure 4-13: Graphic User Interface

4.5 Implementation using JSPASM Package

We divided our system into several processes. Each process identifies an activity or service. Here, we exhibit the state machine of each process separately.

4.5.1 On/Off function

We proposed to create a process for an On/Off operation. This process permits to turn on or turn off the TV. We split the process into three functions.

- The function PowerbuttonEntity.java defines the different transitions that can occur when the process is running.
- The function PowerButtonModel.java states the rules that should be followed.
- The function ON_OFF_engine.java performs the action when the user clicks on the button.

The following is the function for defining the transitions.

```
Import com.codexombie.jspasm.AbsEntity;
Public class PowerButtonEntity extends AbsEntity {
      Public void offAction() {
            wake_up wake_up = new wake_up();
            Main_page main_page = new Main_page();
            Movies movies = new Movies();
            wake_up.setVisible(false);
            main_page.setVisible(false);
            movies.setVisible(false);
            }
      Public void onAction() {
            Welcome welcome = new Welcome();
            welcome.setVisible(true);
            ON_OFF_page on_off_page = new ON_OFF_page();
            on_off_page.setVisible(false);
      }
}
```

The second function for PowerButtonModel.java is shown here

```
import com.codexombie.jspasm.Model;
public class PowerButtonModel extends Model
{
  public PowerButtonModel(String name) throws Exception
  {
            super(name);
            //  Define the entity class
            setEntityClass(PowerButtonEntity.class);
            // Define states
            addState("Off", "offAction", "");
            addState("On", "onAction", "");
            // Set the initial state for new entities
            setInitialState("Off");
            // Define events
            addEventSpec("power", "");
            // Define transitions
            addNormalTransition("Off","power", "On");
  }
}
```

The third function for the action performance is depicted in the code below

```
import java.awt.event.ActionListener;
import com.codexombie.jspasm.Engine;
import com.codexombie.jspasm.Model;
import java.awt.event.ActionEvent;
import java.awt.event.ActionListener;
public class ON_OFF_engine
{
  Public static void main(String[] args) throws Exception
  {
        ON_OFF_page on_off_page = new ON_OFF_page();
        on_off_page.power.addActionListener(new ActionListener(){
          public void actionPerformed(ActionEvent event){
            try{
              //Create a model
              // Create a model
```

```
                    Model PowerButtonModel = new PowerButtonModel("MyModel");
                    // Create an engine
                     Engine engine = new Engine("My Engine");
                    // Add the model
                    engine.addModel(PowerButtonModel);
                            // Start the engine
                     engine.start();
                     // Now we can create an entity
                     engine.createEntity("MyModel", "MyEntity", new Object[0]);
                     // engine.generateEvent("MyEntity", "power");
                     //engine.generateEvent("MyEntity", "power");
                     // Push the button a few times
                     engine.generateEvent("MyEntity", "power");
                     // Wait a couple of seconds
                     Thread.sleep(2000);
                     // Now shut the engine down
                      engine.shutdown();
                     engine.join();
                                     }
                                catch(Exception e){
                                                }
      }
   });
         }
}
```

To switch between pages, we used some of the functions demonstrated before. For instance, when the user wants to navigate on the main page, going back and forth, an OK button has been created to ensure this functionality.

4.5.2 Movie service

To enable video streaming from the database, we developed a small media player. These movie readers give the user the option to choose the desired movie.

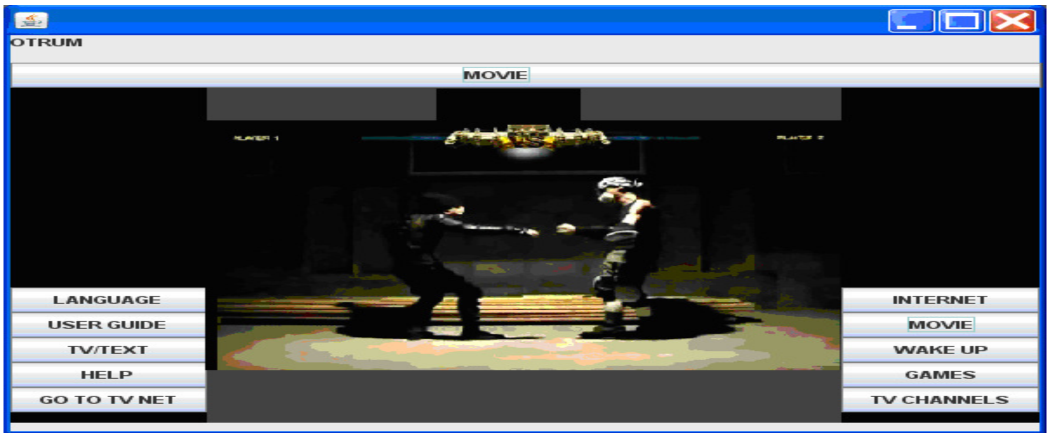

Figure 4-14: Graphic User Interface

4.5.3 Wake Up service

The user can set a wake up timer by going to wake page. We created a process based on the workflow concept to manage this process.

4.5.4 Welcome service

When turning on the TV, a welcome page shows up. In fact, the welcome process is partly related to On/off process. Since the result of this latter is the appearance of a welcome page. Behind the welcome process, some functions that take care of updates and management are performed.

4.5.5 Remote Interface

All mentioned actions are executed through a remote control.
As a solution, we suggested to face the real-world example of a remote interface. This interface involves the appropriate button for the correspondent service.

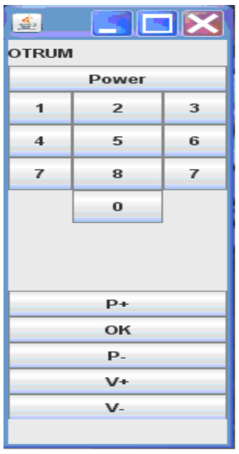

Figure 4-14: remote control screenshot

Chapter 5 Results and Discussion

The main purpose of this project, as we cited before, is to pick the convenient solution for a suitable workflow engine for the OTRUM system. We have showed several approaches in the previous sections. We decided to use the JSPASM package together with Java eclipse in order to build some simple examples of how it can be commonly used. In this section, we will present the way in which we proceed to implement some scenarios listed above and how they can be managed through a workflow engine.

5.1 Results

To compare the performance of each solution with the rest, we did some testing in regards to specific characteristics, like simplicity. We will first try to look at their documentations in order to find out the features of each one. Then, we assign a percentage of simplicity, scalability and adaptability for each workflow engine. We proceed in a subjective way. While doing this, we considered our own opinions when evaluating these products. Afterwards, we judged them in relation to:

- ✓ Time processor
- ✓ Memory consuming
- ✓ Branch size

The chart below represents a comparison between the three selected workflow engines by using simplicity, scalability and adaptability as criteria

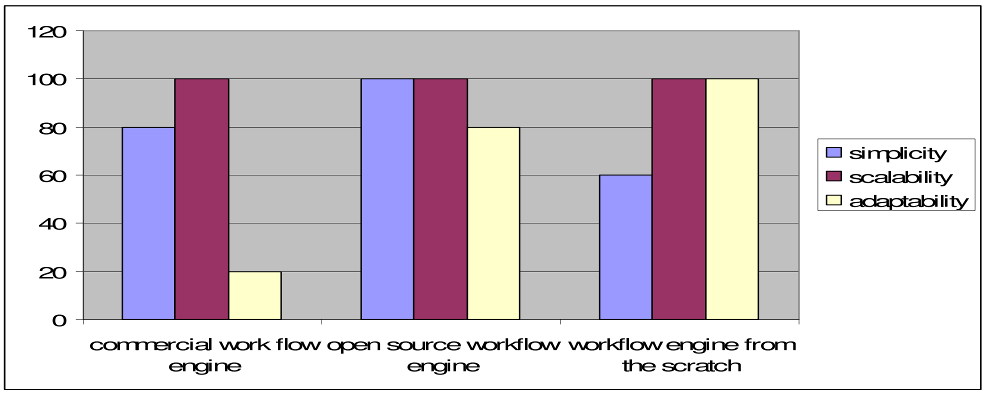

Chart 5-1: Comparison regarding simplicity, scalability and adaptability

The next chart presents the results of our simulation of the three solutions using Time processor and memory consumption as criteria.

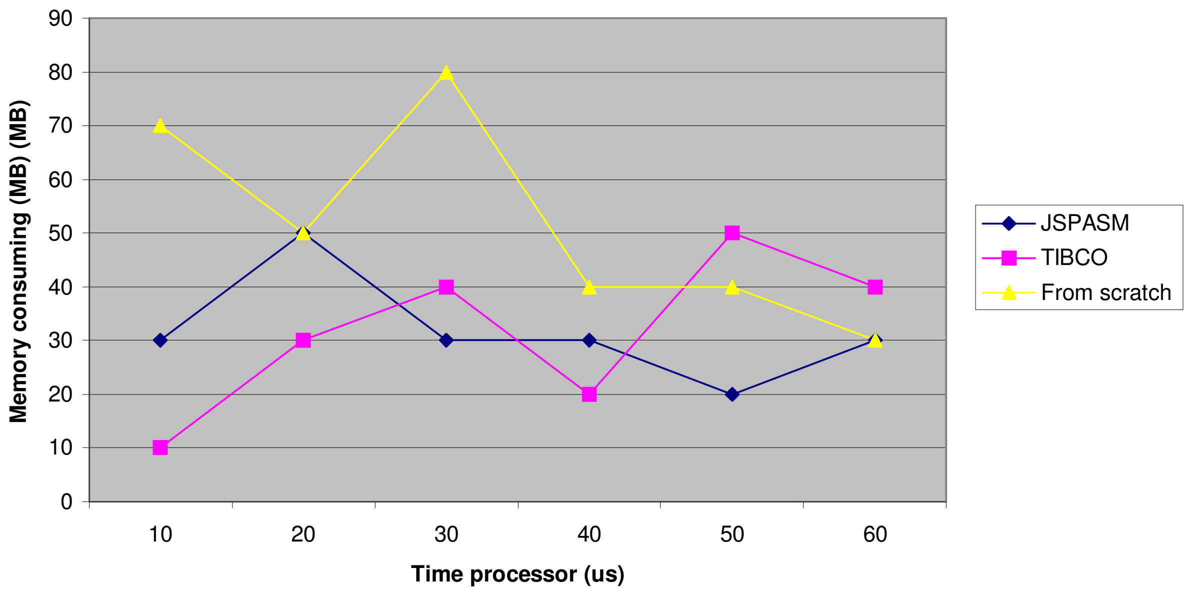

Chart 5-2: The proportionality of Time processor and Memory consuming

As seen in the above chart, the JSPASM solution looks more stable and does not consume memory during the execution. On the contrary, inventing a workflow engine from scratch could be memory consuming, and it occupies almost the whole system resources.

5.2 Discussion

In this section, we will compare the obtained results with our requirements. We will mention the advantages and disadvantages of some of the tests that we conducted using different solutions, characterizing the commercial workflow engines.

In a commercial workflow engine, several products can be found. To assess the performance of the available workflow engines, we take several features into consideration, including scalability, adaptability, interoperability, and simplicity. The first workflow engine tested was iMarkup. It is simple to use, but is non-scalable, non-interoperable, and non-adaptable. For these reasons, iMarkup was rejected. The second workflow engine tested was Microsoft's Biz Talk Server, which we rejected due to its poor adaptability and non-interoperability. The Oracle Business process manager was also rejected due to its bad scalability and its complexity. Finally, it was decided to stick with the TIBCO workflow due to its good scalability, besides its adaptability, interoperability, and simplicity. A commercial workflow engine can respond to the needs of the OTRUM system. In the following paragraphs we will be listing the advantages and disadvantages of commercial workflow engines.

5.2.1 Advantages

The first advantage of the TIBCO workflow engine resides in the fact that they offer a graphical user interface, which makes the workflow engine user friendly. In addition, the graphical user interface allows for a dynamic construction of workflows. For instance, the transition between different states in workflow can be based on probabilistic factors. This fact makes the execution of the workflow differ from one execution to another depending on given conditions. It is also an advantage that the TIBCO workflow engine allows for easy construction of a specific workflow, this without the need to construct a workflow engine from scratch.

5.2.2 Disadvantages

One of the biggest disadvantages of a commercial workflow engine is its high cost. In addition, OTRUM will become dependent to the workflow engine provider. In case that an updated version of the workflow engine is issued, OTRUM will need to purchase a new licence. Furthermore, the company could loose its confidentiality. Moreover, when opting for a commercial workflow engine, the company needs to adapt it to its own system. The adaptation task is not trivial.

In the following figure, we draw the principle adaptation of this solution.

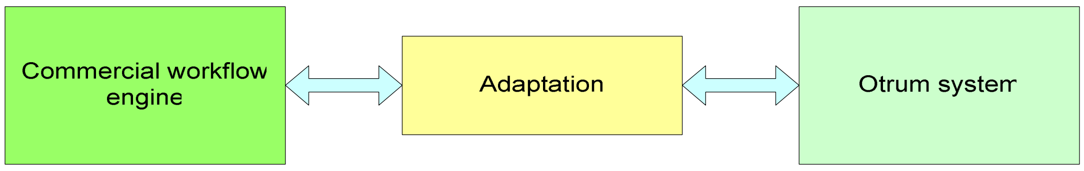

Figure 5-1: Integration of a commercial workflow engine in OTRUM system

5.3 Characterization of an open source workflow engine

An open source workflow engine could be efficient and convenient. Several open source workflow engines could be found. To evaluate different open source workflow engines, an audit is done based on their scalability, adaptability, interoperability, and simplicity. The first considered open source workflow engine is Spring which, is scalable and interoperable. However, Spring is non adaptable and complex. For these reasons, Spring was rejected. Twister was rejected due to its complexity and non-scalability. JBPM workflow engine is non-adaptable and non-scalable. The WfMOpen is also rejected since it is non-interoperable and non-scalable. The JWT has very good features with regards to scalability, adaptability, interoperability, and simplicity. Unfortunately, the JWT workflow engine is still under development and thus not ready to use. The JSPASM has very good features with respect to scalability, adaptability, interoperability, and simplicity. As consequence, we picked up JSPASM package for Java eclipse so that we can easily integrate it into our system. This solution provides us all possible features to build a good workflow engine.

5.3.1 Advantages

The JSPASM workflow engine is free and efficient. Furthermore it allows for individual processing and state handling for each separate TV-set in the hotel network. In addition, JSPASM is characterized by its simplicity and its ability to be extended which allow OTRUM system to be flexible. JSPASM is 100% pure Java package for eclipse. Moreover, JSPASM allows for handling multiple state models in parallel, with multiple entities operating in parallel within each model. JSPASM is able to hold information form a key press event about what TV-set event is received from and which key on the Remote control has been pressed. Furthermore, JSPASM is able to collaborate smoothly with new OTRUM core framework modules using Java VM.
A new plug-in for Java eclipse, is currently in development, and will soon be available. The figure below presents the characteristics of an open source workflow engine.

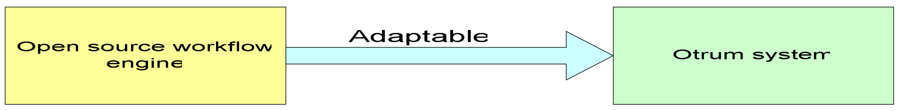

Figure 5-2: Integration of an open source workflow engine in OTRUM system

5.3.2 Disadvantages

For our simple scenario and testing, an open workflow engine based on JSPASM Java package presents no real disadvantages.

5.4 Characterization of a workflow engine from the scratch

This solution in connection with an open source workflow engine has many advantages and drawbacks. First of all, we assumed that this solution should be based on the Java environment. The main advantage of this solution is the scalability, i.e. to extend the system or to add new features. It is easy to incorporate them into a workflow engine built from the scratch. Nevertheless, this solution could not be as good as other workflow engines. It takes time for the company to develop its own workflow engine, and no guarantee for the efficiency it will provide once completed. In addition, there is little need for inventing one from scratch, as there are already several open source workflow engines available that can achieve our goals. For this reason, we proposed using an open source workflow engine and exploited its features in favour of our requirements. Below, we show a possible way to create a workflow engine from the scratch and how it could be connected to the OTRUM system.

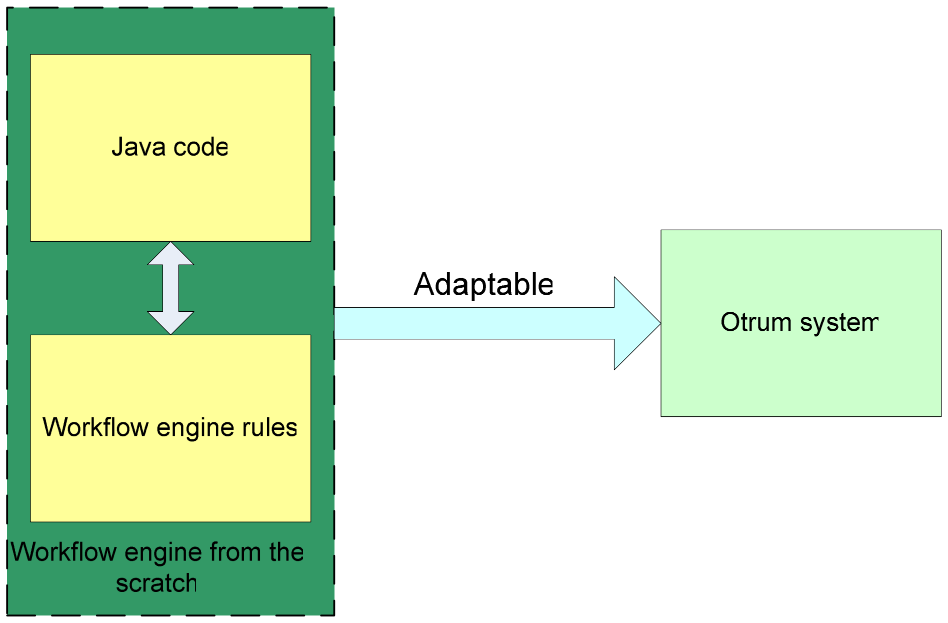

Figure 5-3: integration of an open source workflow engine in OTRUM system

Chapter 6 Conclusion

In this thesis, we discussed and implemented a suitable workflow engine solution. An interactive TV workflow engine based on JSPASM package was built and tested. Starting from three proposed solutions such as commercial workflow engine, open source workflow engine and a workflow engine created from the scratch, we discussed and evaluated the features of each one.

We began with researching “state-of-the-art” technology related to workflow engine solutions. It is important for us to specify our requirements in order to select the best solution.

To approve our concept and research the possibilities of workflow engine technology, we developed a small example dealing with the OTRUM system. It will be a reference for us to judge the performance of our approach at the regard of other solutions.

Next, we sketched some common scenarios for testing services that the system must provide. We pointed out 4 possible scenarios: a greeting scenario, a wake-up scenario, a movie scenario and a remote control scenario. We compared the performance of the three solutions and figured out the main advantages and disadvantages of each.

To integrate these services under a workflow engine, we created a state machine process for each service. In the future, with the emergency of all IP services, the interactive TV can still exist and expand with new useful services. How to interoperate the existing OTRUM system with the new system is very important for OTRUM, but not addressed sufficiently in the literature.

Our thesis proposed and evaluated three workflow engine solutions:

- a commercial workflow engine e.g. TIBCO Business Work
- an open source workflow engine e.g. JSPASM Java package
- a workflow engine raised from the scratch under Java environment

Our solutions can be considered as a testing example for the implementation of a workflow engine for interactive TV. We proved that an existing open source workflow engine is sufficient, and that it could fulfill our goals and requirements.

References

[1] A. Paventhan, K. Takeda, S.J. Cox, and D.A. Nicole. "Leveraging Windows Workflow Foundation for Scientific Workflows in Wind Tunnel Applications", IEEE Workshop on Workflow and Data Flow for Scientific Applications (SciFlow'06) , Atlanta, GA.

[2] I. Altintas, C. Berkley, E. Jaeger, M. Jones, B. Ludäscher, S. Mock, "Kepler: An Extensible System for Design and Execution of Scientific Workflows", 16th Intl. Conf. on Scientific and Statistical Database Management (SSDBM'04)

[3] Omer Rana, "Creating and Managing Distributed Scientific Workflows: Techniques and Tools", tutorial, EuroPAR 2005 Conference

[4] I. Foster, J. Voeckler, M. Wilde, Y. Zhao, "Chimera: A Virtual Data System for Representing, Querying, and Automating Data Derivation", 14th International

[5] Jia Yu and Rajkumar Buyya, "A Taxonomy of Workflow Management Systems Computing", Technical Report, GRIDS-TR-2005-1,Computing and Distributed Systems Laboratory,
University of Melbourne, Australia, March 10, 2005.

[6] Shalil Majithia, Matthew S. Shields, Ian J. Taylor, and Ian Wang, "Triana: A Graphical Web Service Composition and Execution Toolkit", Proceedings of the IEEE International Conference on Web Services
(ICWS'04), pages 514-524. IEEE Computer Society,

[7] Ali Shaikh Ali, Omer F. Rana and Ian J. Taylor, "Web Services Composition for Distributed Data Mining", Workshop on Web and Workflow Services for Scientific Data Analysis (WAGSSDA), Oslo, Norway, 2005.

[8] http://www.otrum.com/

[9] Broad, William J. The Oracle: Ancient Delphi and the Science behind its Lost Secrets, 2006. ISBN 1-59-420081-5.

[10] Mike Havey, "Essential Business Process Modeling", 'Chapter Seven. The Workflow Management Coalition (WfMC)', O'Reilly, August 2005. ISBN 0-596-00843-0.

[11] Douglas E. Comer, "Internetworking with TCP/IP", Chapter eleven. User Datagram Protocol (UDP), Mai 2006. ISBN 0-13-187671-6

[12] Birrer, A. and T. Eggenschwiler (1993). Frameworks in the financial engineering domain: an experience report. In (eds), Springer-Verlag, proceedings of the European conference on object-oriented programming, Kaiserslautern, Germany: 21-35.

[13] Y.Daniel Liang Introduction to Java Programming-Comprehensive Version (6th Edition) (Paperback)

[14] http://www.imarkup.com/

[15] http://www.microsoft.com/biztalk/default.mspx

[16] http://msdn2.microsoft.com/en-us/netframework/aa663328.aspx

[17] http://www.tibco.com/software/application_integration/businessworks/default.jsp

[18] http://www.oracle.com/technology/bpel/index.html

[19] http://www.javaworld.com/javaworld/jw-04-2005/jw-0411-spring.html

[20] http://jspasm.sourceforge.net/

[21] http://java-source.net/open-source/workflow-engines/twister

[22] http://www.jboss.com/products/jbpm

[23] http://wfmopen.sourceforge.net/

[24] http://www.manageability.org/blog/stuff/workflow_in_java/view

[25] http://www.eclipse.org/proposals/jwt/

[26] http://jspasm.sourceforge.net/ ,http://jspasm.sourceforge.net/jspasm.html , http://www.eclipse.org/proposals/jwt/

Abbreviations

UDP	User Datagram Protocol
IP	Internet Protocol
IDEs	Integrated Development Environments
ALOHA	Instance for ODM Fusion Driver (OTRUM)
API	Application Programming Interface
GNU	not UNIX
IBM	International Business Machines Corporation
WfMOpen	Workflow Management open
J2EE	Java 2 Platform, Enterprise Edition
WfMC	Workflow Management Coalition
XML	Extensible Markup Language
XPDL	Procès Définition Lagunage *ou* langage
JWT	Java Workflow Toolbox
WE	Workflow Editor
WAM	Administration and Monitoring
JBPM	java Business Process Mgmt
SDL	Specification and Description Language