



TAMPEREEN TEKNILLINEN YLIOPISTO
TAMPERE UNIVERSITY OF TECHNOLOGY

JUHA LAUTTAMUS

AN ORCHESTRATION PROCESS OF ANALYTIC SERVICES IN
HOLISTIC ENERGY MANAGEMENT SYSTEMS

Master of Science Thesis

Examiner:
Prof. Jose L.Martinez Lastra
Examiner and topic approved by the
Council of Engineering Sciences on
4 December 2013

ABSTRACT

TAMPERE UNIVERSITY OF TECHNOLOGY

Master's Degree Programme in Automation Engineering

LAUTTAMUS, JUHA: An Orchestration Process of Analytic Services in Holistic Energy Management Systems

Master of Science Thesis, 79 pages, 17 Appendix pages

May 2015

Major: Factory Automation

Examiner: Prof. Jose L. Martinez Lastra

Keywords: Energy Management System, Service-Oriented Architecture, Business Process Management, Analytic tools

Energy Management System (EMS) is a concept that is an essential part of modern manufacturing enterprises. The goal of EMS is to offer the surrounding systems with decision-support and control tools based on analytic operations that allow the optimization of the energy usage. This thesis presents an orchestration process of analytic services that enables the effective management of analytic operations within EMS.

The current transition from Internet of People towards Internet of Things is expected to significantly increase the amount of available energy-related information. This will increase the level of complexity of the required analytic tools. In order to manage the increasing complexity the Service-Oriented Architecture (SOA) is utilized in the orchestration process, allowing the flexible organization and rapid deployment of new analytic functionality.

The thesis work is divided into two parts. The literature review part studies the current state of research in EMS and the related analytics. Weight is also put on studying of the research attempting to acquire holistic EMS solutions. Holistic EMS targets to manage the energy consumption of the whole system in a way that considers the specific requirements of each sub-system.

In the implementation part a variety of Internet-based technologies are applied to provide an implementation of the orchestration process of analytic services. An Enterprise Service Bus is used as a platform for the implementation, supporting the integration of systems. The implementation is used to demonstrate the capabilities offered by the orchestration of analytic services.

The results of this thesis indicate that the service-based approach increases the manageability of the analytic operations in EMS. The solution allows the rapid development of new analytic processes from location-independent analytic services. The research leading to these results was partially funded by the European Union Seventh Framework Programme (FP7/2007-2013) under grant agreement n° 600058.

TIIVISTELMÄ

TAMPEREEN TEKNILLINEN YLIOPISTO

Automaatiotekniikan diplomi-insinöörin tutkinto

LAUTTAMUS, JUHA: An Orchestration Process of Analytic Services in Holistic Energy Management Systems

Diplomityö, 79 sivua, 17 liitesivua

Toukokuu 2015

Pääaine: Factory Automation

Tarkastaja: Prof. Jose L. Martinez Lastra

Avainsanat: energianhallintajärjestelmät, palvelupohjainen arkkitehtuuri, liike-prosessien hallinta, analyyttiset työkalut

Energianhallintajärjestelmät ovat olennainen osa nykyaikaisia tuotantojärjestelmiä. Energianhallintajärjestelmien tarkoitus on tarjota ympäröiville järjestelmille analyyttisiin operaatioihin perustuvia päätöksentekoa tukevia toimintoja ja työkaluja, jotka mahdollistavat energian kulutuksen optimoimisen. Tämä diplomityö esittää analyyttisten palvelujen orkestrointiprosessin, joka mahdollistaa analyyttisten työkalujen tehokkaan hallinnan energianhallintajärjestelmissä.

Meneillään olevan siirtymisen ihmisten Internetistä (Internet of People) tavaroiden Internetiin (Internet of Things) odotetaan huomattavasti lisäävän tarjolla olevaa energiaan liittyvän tiedon määrää, mikä lisää tarvittavien analyyttisten työkalujen monimutkaisuutta. Tässä diplomityössä hyödynnetään palvelupohjaista arkkitehtuuria (Service-Oriented Architecture) tämän monimutkaisuuden hallitsemiseksi. Tuloksena on orkestrointiprosessi, mikä mahdollistaa uuden analyyttisen toiminnallisuuden joustavan jäsentelyn ja nopean käyttöönoton.

Tämä diplomityö on jaettu kahteen osaan. Ensimmäisessä osassa suoritetaan katselmus tämänhetkisistä energianhallintajärjestelmistä ja niiden analyyttisistä työkaluista. Tärkeänä osa-alueena keskitytään holistisiin energianhallintajärjestelmiin. Holistiset energianhallintajärjestelmät pyrkivät kokonaisvaltaisiin ratkaisuihin, jotka keskittyvät kokonaisuksiin alijärjestelmien sijaan.

Toisessa osassa esitetään analyyttisten palveluiden orkestrointiprosessin toteutus, joka pohjautuu Internet-pohjaisiin teknologioihin. Toteutuksen alustana käytetään yrityspalveluväylää, mikä tukee järjestelmäintegraatiota. Toteutuksen avulla esitetään analyyttisten palveluiden orkestrointiprosessin tarjoamia etuja.

Tämän diplomityön tulokset osoittavat, että palvelupohjainen lähestymistapa lisää analyyttisten prosessien hallittavuutta tuotantoyritysten energianhallintajärjestelmissä. Orkestrointiprosessi mahdollistaa uusien analyyttisten prosessien nopean kehityksen ja käyttöönoton. Näihin tuloksiin johtanutta tutkimusta rahoitettiin osittain Euroopan Unionin 7. puiteohjelman (FP7/2007-2013) toimesta apurahasopimuksella nro. 600058.

PREFACE

This thesis work was made in the FAST-Lab. at Tampere University of Technology. The research leading to these results was partially funded by the European Union Seventh Framework Programme (FP7/2007-2013) under grant agreement n° 600058”.

Now that the work is done, I would like to express my gratitude to Professor Jose L. Martinez Lastra for the support and opportunity offered to me to complete my thesis in the FAST-Lab. The premises, assets and the knowledge provided by the laboratory and its personnel were greatly inspiring, and gave me a chance to grow as an engineer.

To the supervisor of my thesis, Anna Florea, I want to address my sincere appreciation. Your guidance and patience enabled the results of this thesis. I want to thank also my colleagues Anton and Muhammad who worked in the URB-Grade project, and wish you all the best of luck in the future.

Finally, I want to thank Laura and my family for their endless support and patience.

Jyväskylä, May 20th, 2015

Juha Luttamus

TABLE OF CONTENTS

Abstract	i
Tiivistelmä	ii
Preface.....	iii
1. Introduction	1
1.1 Background	1
1.2 Problem Definition.....	3
1.2.1 Justification of the Work.....	3
1.3 Work Description	4
1.3.1 Objectives	4
1.3.2 Methodology	4
1.3.3 Scope and Limitations	5
1.3.4 Thesis Outline	5
2. Literature and Technology Review	6
2.1 Review on Energy Management Systems.....	6
2.1.1 Current Trends in Energy Management Systems	6
2.1.2 Holistic Energy Management Systems	8
2.2 Analytics in Energy Management Systems.....	9
2.2.1 Analytic Tools in Energy Management	10
2.2.2 Review on Data Mining in Manufacturing Systems.....	12
2.2.3 A Case of Study: URB-Grade.....	13
2.3 Service-Oriented Computing	15
2.3.1 SOA in the Manufacturing ICT Systems.....	15
2.3.2 Enabling the SOA with Enterprise Service Bus	16
3. Methodology	18
3.1 An Orchestration Process of Analytic Services	18
3.1.1 Conceptual Architecture	18
3.1.2 Quality of Analytic Services.....	19
3.2 Service-Oriented Architecture	21
3.2.1 Web Services	21
3.2.2 SOAP	22
3.2.3 Java Technologies for Web Services	23
3.3 Platform for SOA	23
3.3.1 OSGi Component Model	23
3.3.2 Apache ServiceMix.....	25
3.4 Management of Analytic Processes	26
3.4.1 BPMN 2.0	26
3.4.2 jBPM workflow suite.....	27
3.5 Data handling in Analytic Processes.....	28
3.5.1 E-Nodes	28
3.6 Other Software Tools	29

3.6.1	Maven	29
3.6.2	Spring.....	30
3.6.3	Mojarra JavaServer Faces.....	30
4.	Implementation	31
4.1	System Architecture and Components.....	31
4.1.1	Interactions Between System Components.....	32
4.1.2	Analytic Manager OSGi Bundle.....	33
4.1.3	Analytic Service OSGi Bundles	36
4.1.4	Ontology Module Web Service	38
4.1.5	Client UI	40
4.2	Analytic Service Orchestration	41
4.2.1	Injection of Analytic Process Definitions.....	41
4.2.2	The Use of the OSGi Service Registry	44
4.2.3	Invocation of Analytic Services.....	45
4.2.4	Data structure for Analytic Data.....	47
4.3	Software Project Organization	48
5.	Results.....	49
5.1	System Configuration	49
5.2	Applied Orchestration Process.....	49
5.3	Analytic Service Set and Service Tasks.....	50
5.4	Implemented Analytic Processes	51
5.5	Demonstration of Analytic Processes	53
5.5.1	Sample Data for Analytic Processes.....	53
5.5.2	Single Analytic Service	54
5.5.3	Parallel Analytic Services.....	57
5.5.4	Series of Analytic Services.....	59
5.5.5	Demonstration of a User Decision.....	62
5.5.6	Demonstration of a Process Decision.....	64
5.5.7	Demonstration of Service Selection	69
6.	Conclusion	72
6.1	Future Work	72
	References.....	74
	Appendix A: System configuration.....	80
	Appendix B: XML interfaces.....	81
	Appendix C: Package diagram.....	97

1. INTRODUCTION

The energy is the driver of the modern society. In its different forms energy is the potential that provides us with movement, heat, light and power; everything that is required for everyday living and working. However, energy is a limited resource and its utilization is not free; extensive amount of resources such as work, money and research are constantly required to manage the transformation process of energy into its different forms. Environmental effects are also significant due to emissions and required raw materials.

Manufacturing industry is an energy-intensive field, where energy efficiency is of high importance. The manufacturing companies need to optimize their energy consumption in order to withstand in the markets and to reach the international emission reduction goals [1]. In order to optimize the consumption various analytic tools are needed, that allow the effective monitoring and controlling of available energy-related information.

The optimization of energy usage requires awareness. Due to the complexity of energy processes the information needs to be gathered from different sources and combined. Timely information is also needed as the energy processes can be dynamic by nature. Here ICT-based (Information and Communications Technology) solutions prove to be useful because of the capability to process huge amounts of information and networking.

Also the technological change from Internet of People (IoP) towards Internet of Things (IoT) is changing the field of manufacturing. It is expected that the amount of available energy-related information will rapidly increase due to the emerging of IoT. Increase in information will allow more effective monitoring and control of energy resources. As a prerequisite it is required to have effective means of managing the analytic operations in the energy management.

1.1 Background

Energy Management System (EMS) is a general term for software that effectively monitors and controls energy consumption and generation. EMS is applied in industrial, commercial and residential sectors in order to decrease the expenses and emissions, and to allow more optimized use of energy resources. Therefore the responsibilities set for EMS can be divided into two categories that are the optimization of energy usage and the increasing of energy-awareness in the system.

Traditionally manufacturing sector has concentrated on improving the cost-effective productivity as the main priority, but the rising energy prices and the global markets force manufacturing companies to organize their operations in an energy-efficient manner.

In manufacturing the EMS acts as an underlying system used by other systems, such as ERP (Enterprise Resource Planning), MES (Manufacturing Execution System) and plant floor. EMS provides its users with visualizations and illustrations of data that increase their knowledge of the company's energy consumption. The information can be used to recognize the performance levels of different operations to recognize the potential targets of upgrade. EMS may also provide means for finding of reasons behind machine failures and the prediction of the future ones. [2]

The amount of available data in manufacturing systems will rapidly increase due to the expected transition from IoP into IoT before 2020 [3]. More and more various appliances will be producing different kinds of information that can significantly improve the common energy awareness. Large data sets containing diverse data require sophisticated analytic means such as cloud computing that provides efficient resource management for data storing and mining [4].

Rarely the raw data itself can provide meaningful information. A necessary part of the data acquisition is the data aggregation that prepares raw data for use. Key-performance indicators (KPI) are higher level information aggregated from the raw data in a way suitable for the business case. They can expressively describe the performance of various systems. Building analytic operations onto KPIs allows the concentration on aspects important for the current domain.

The progress in modern IT systems pushes towards service-oriented architectures acting over Internet. In such systems enterprise's functionality and operations are organized into a reusable set of services. On top of the services it is possible to quickly compose new functionality. This progress emerges new markets around the services. The enterprises do not need to master operations outside their key competence as the service oriented architectures allow the aggregation of externally managed services.

The changing business environment increases the amount of responsibilities set for EMS. The energy-awareness within manufacturing enterprises increase and the scope of energy-management grows from only managing the manufacturing processes. Holistic EMS needs to be able to manage the energy aspects of the whole enterprise, including all of the functions and infrastructure that depend on energy.

Part of the holistic approach is also the different user groups and their requirements within an enterprise. Different users have different needs regarding the information and knowledge of the assets they work on. In order to provide value for all the users, EMS needs to support their perspectives into the domain.

1.2 Problem Definition

In the field of manufacturing the EMS analytic operations are significantly heterogeneous. This is caused by the varying nature of manufacturing processes that require very specific combinations of analytic algorithms and configurations. Sporadic analytic operations are inefficient and difficult to manage and implementation of widely adoptable analytic operations is difficult.

The requirement for efficient management of analytic operations is also underlined by the transition to IoT. Large amounts of various kinds of information will become available and make the required analytic operations more complex. Therefore the traditional time-consuming approach of creating case-specific analytic operations may not be adequate.

Another problem relates to the complexity of the energy processes. They can rarely be isolated into bare manufacturing processes. Instead, holistic approaches should be considered. Holistic energy management enables the energy optimization within the whole enterprise, which outperforms the non-holistic approaches by the potential benefits.

Finally, modern enterprise systems are based on integration. The enterprise operations are built on integrated systems. In order to utilize the effectively utilize the energy management tools, it has to possible to integrate the energy management tools to the existing enterprise systems.

This thesis attempts to answer i.a. the following questions: *What are the steps of the orchestration process of analytic services? How to manage the analytic operations in EMS? How to facilitate the holism in EMS analytics? How to integrate EMS into manufacturing enterprise systems?*

1.2.1 Justification of the Work

The problems stated in the previous section are significant for the energy management systems of manufacturing industry in the time of big data and distributed systems. This thesis attempts to provide various benefits to the EMS design in the field of manufacturing. The management of the analytic processes in service-oriented manner allows the companies to quickly introduce new analytic processes to provide means of decision support for their enterprise systems.

The progress towards more distributed IT systems also makes the organization of business processes distributed. For companies it becomes more favorable to build business processes by utilizing SOA. The approach given in this thesis presents a way to organize the analytic processes of EMS regarding the SOA. The approach grants the solution a high level of flexibility as SOA permits the solution to be integrated into different EMS as a service.

Holistic approaches are those that consider systems in their entirety rather than just focusing on specific properties or specific components. Therefore holistic EMS can be expected to provide greater benefits than EMS with limited scope.

1.3 Work Description

This section describes the objectives, the methodology, the scope and limitations, and the outline of this thesis.

1.3.1 Objectives

The main objective of this thesis is to allow the rapid development of analytic processes in EMS. The completion of this objective is expected to provide a solution to the problems defined in the section 1.2.

The main objective can be broken down into following sub-objectives, whose completion leads to the completion of the main objective:

1. Find ways to manage the complexity of analytic operations in EMS. The complexity is caused by the sporadic case-specific requirements of manufacturing EMS and the growing amounts of available data provided by the transition into IoT.
2. Search for means that enable holistic energy management. Holistic energy management tools are needed in order to enable effective energy management in modern manufacturing enterprises. The energy-related processes of manufacturing enterprises are spread between various domains, which makes holism a prerequisite of complete energy management.
3. Design an architecture for the solution that supports integration. It is essential that the solution can be integrated into existing manufacturing ICT systems, allowing its effective utilization.
4. Demonstrate the capabilities of the approach to evaluate its feasibility. On base of the results it is possible to estimate if the solution is applicable in real manufacturing environments.

1.3.2 Methodology

In order to achieve the objectives of this thesis, following steps are taken:

Literature review

A study is made on the current trends and requirements of EMS and their analytic operations. Specific interest is given on approaches that embrace holism. Information is collected on how the analytic service orchestration should be designed to make it applicable in modern EMS and manufacturing enterprise ICT systems.

An orchestration process of analytic services

An orchestration process of analytic services is described including an overall architecture for the solution. The logical orchestration process shall allow the composition of

analytic services into analytic processes. A set of software tools are selected that can be applied to implement the orchestration process.

Application of the orchestration process

An implementation of the orchestration process is developed. The resulting tool is applied to demonstrate different use cases of analytic processes with a set of analytic services. The solution shall be based on web technologies and a modern integration platform.

1.3.3 Scope and Limitations

The approach presented in this thesis is targeted to be used only in the domain of manufacturing industry. Different fields and industries have their own requirements for the energy management and therefore they may not be supported.

The information aggregated in key performance indicators (KPI) and raw data that are processed within the orchestrated processes are managed in its own module that is outside the scope of this thesis. The mapping between perspectives and data is also outside the scope of this thesis. The solution is not planned to operate in a real time manner.

Certain restrictions exist in the solution: the system doesn't function in real-time, nor operate real-time data; information security (i.e. user authentication and authorization) is not considered.

1.3.4 Thesis Outline

This thesis work is organized as follows. This section included the problem definition and the approach to solve the presented problem. Chapter 2 concentrates on the state of the art of the research and technologies in EMS analytics. Chapter 3 presents the methodology that is used to solve the problem. Chapter 4 presents an implementation where the methodologies were applied that demonstrates the capabilities of the solution. Chapter 5 presents the gained results and reflects them to the set goals of the thesis. Chapter 6 presents the overall conclusion of this thesis and overviews the future research.

2. LITERATURE AND TECHNOLOGY REVIEW

This section reviews the current status of the research and technology related to the Energy Management Systems and their analytic operations.

2.1 Review on Energy Management Systems

This section reviews the recent research targets in Energy Management Systems and the means used to achieve holistic EMS solutions.

2.1.1 Current Trends in Energy Management Systems

New types of EMS have been emerging rapidly in the last few years as a response to the market needs and the emerging technologies. The areas of appliances range to various multidisciplinary fields of research including proposals for managing energy consumption in contexts such as buildings [5], homes [6], manufacturing [7], urban infrastructure [8] and cloud-based ICT [9]. Current appearing trends seem to pinpoint into the integration of systems via architectures based on SOA, big data, cloud services and wireless sensor networks (WSN) as stated in the following sections.

The field of manufacturing has been a significant research target for applying the EMS due to the energy intensity of the manufacturing processes [10]. In manufacturing EMS provides means to lower the energy consumption and the amount of the wasted raw materials, improve the product traceability to avoid the production line stoppages, and also to enhance machinery management in order to reduce the energy consumption during the manufacturing process.

In the close future Internet of Things is expected to alter the field of EMS. IoT provides devices with digital identities and simplifies the communication with them [11]. Transition into IoT will cause a significant increase in the amount of available measured data and allows a more thorough energy management, provided that the EMS is granted with the capabilities to process large amounts of diverse data.

Common characteristics exist in the modern EMS targeted for manufacturing industry. In [12] it is stated that the main enabler of the energy awareness is achieved through the integration of systems. The research uses the classic ISA-95 standard with its definition of an architectural model for automation systems as its guideline. ISA-95 architecture is presented in Figure 1. ISA-95 defines the following layers: Enterprise Resource Planning (ERP) layer, Manufacturing Execution System (MES) layer, and control, field and process layers [13]. Each layer operates with different functionalities, requirements, time scales, technologies and data, setting borders of communication between the sys-

tems. Finding effective solutions for bypassing these borders enhances the flow of information in manufacturing enterprises. This also enables the integration of EMS into the manufacturing processes, providing optimization in near real-time.

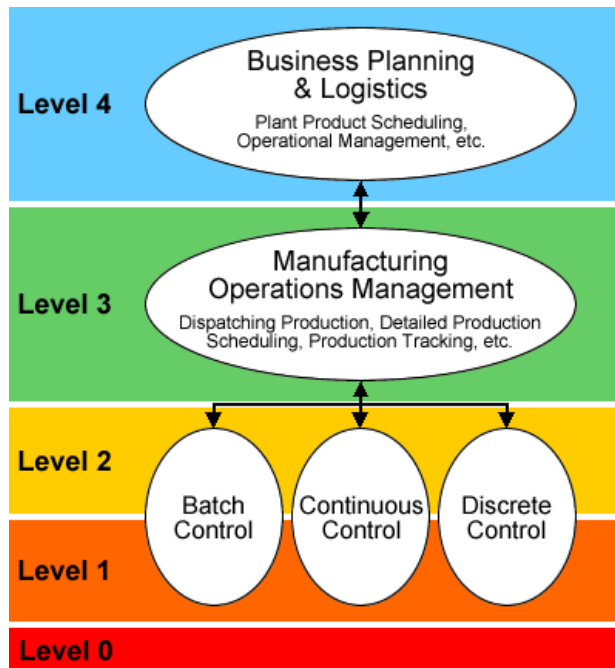


Figure 1: ISA-95 architecture [14]

In a recent PLANTCockpit research project this requirement of integration was also addressed [15]. Architecturally PLANTCockpit applies SOA based on an Enterprise Service Bus (ESB), which provides integration means via common technological interfaces. The presented approach is suitable to be used as a base for implementing loosely-coupled enterprise-scale applications.

In [7] a decision support tool is presented that utilizes the results of the PLANTCockpit project. The system performs operations on KPIs aggregated from different data sources to provide enhanced energy management means to the users of the system. The outcome of the research is expected to improve the energy efficiency and reduce carbon emissions and waste production in manufacturing processes.

Modern manufacturing EMS attempt to provide a pervasive energy information management solution. In [16] an approach is presented where EMS is distributed on both ERP (eEMS) and MES (mEMS) levels of the enterprise. This design decision is planned to enhance the EMS performance as the time scales and data characteristics differ greatly from each other on the separate layers of automation systems. The suggested eEMS and mMES use Internet protocols for communication. The proposed solution produces results and events utilizing KPIs and CEP. The presented solution resembles the event-driven SOA [17].

In [6] a Home Energy Management System (HEMS) is proposed that aims to achieve improvements in energy efficiency. A Wireless Sensor and Actor Network (WSAN) is built to monitor and control the electric sockets. The data is collected

through the sensors and via Internet connection sent to a database. Data mining is applied on the collected data to implement the data analysis. Analysis is done on the continuous flow of data received by the database. More specifically data clustering is used as the method. HEMS uses the acquired criterion knowledge to determine the operation states of the home appliances. The solution is capable of controlling the plugged-in appliances through identification achieved from the measured values of active power, reactive power and current. The given approach provides flexibility and scalability via the sensor network and processing of large amounts of information due to the data mining capabilities.

From the reviewed research a common requirement for effective integration of systems can be seen. The EMS needs to be integrated into the manufacturing subsystems following the ISA-95 specification. Here the EMS is expected to provide decision support for the company's operations. Energy awareness is achieved via modeling of the systems and massive acquisition of energy-related data. Analytic methods are applied in order to optimize the behavior of the system.

2.1.2 Holistic Energy Management Systems

In holistic energy management the energy consumption is not considered solely as the inputs and outputs of the manufacturing processes or single devices, but of the whole company with its assets and employees. In holistic thinking the whole system is greater than the sum of its parts. Holistic EMS allows a manufacturing facility to manage the overall energy usage with increased performance and to recognize the complex relationships between various parts of the system.

The system integration is seen as the key when reaching for holism. The research performed in [7] presents an approach where the holistic energy management is achieved via systems integration. The information of the whole domain, containing the different ISA-95 layers, is aggregated into KPIs that attempt to describe the complex relationships in energy consumption. Finally the KPIs are used as an input to a decision-support system that supports its users to identify improvement opportunities and in predicting the effects of changes. The approach supports real-time control.

In [10] an architecture is presented for implementing holistic EMS that aims to manage the energy usage both within the manufacturing plants ERP and MES, and the building itself. It utilizes eEMS for ERP layer and mEMS providing DSS for Factory Automation Systems (FAS) and Building Automation Systems (BAS). Two tailored EMS are designed to meet the different requirements of the ERP and MES, including the variety in data and data sources, and the time scales. The implementation utilizes complex-event processing with reasoning capabilities. The EMS relies on the use of well-designed KPIs that illustrate the energy efficiency in both manufacturing and building domains. A variety of meters are needed to measure the state of the system.

EMS provides means of finding the key variations yielding the greatest potential for an increase in output or efficiency. The responsibilities of holistic EMS include the capability of efficiently gathering data, establishing links between output and the data and

inspecting the controllable variety. Therefore the required capabilities can be divided to categories such as measuring and organization of data, modeling of the energy related dependencies, analytic processing and presentation of results. [10]. The holistic EMS is needed in order to manage the complexity of the whole manufacturing facility, and therefore directly affects the benefits provided by the EMS.

Various approaches are presented in [12] that attempt to provide improvements to the inter-system communication. Different standards and protocols such as Web Services, SOA and System of Systems are endorsed. By combining these methodologies distributed holistic systems communicating over Internet are pursued. Commonly distributed systems and cloud-based architectures have proved to be beneficial due to the provided technological aspects, i.a. scalability and maintainability, and support in the current markets. Applying the same means to the development of EMS is important in order to integrate EMS to the modern enterprise systems.

In [18] research project targeted for urban areas HEMS is defined as a system that provides a solution with fully interoperable software tools capable of holistic management of energy supply and demand in urban areas. In this case the system serves a group of end-users: *district facilities managers, energy utilities, operators, building managers*, etc. The solution provides them with holistic monitoring and decision-support tools for energy management. This is an important notion as a holistic EMS needs to support different user types of the system with their own perspectives into the domain and its information.

Manufacturing enterprises usually have multiple user perspectives into the enterprise's energy-related assets. Perspectives considering production, enterprise, building and office are common for manufacturing facilities. These perspectives concentrate on different points of interest in the domain data with different requirements for the analytic processes to be executed. In order to have the EMS operating effectively it needs to satisfy users with different perspectives.

Measuring and monitoring of information are essential when reaching for the holistic energy awareness. This need demands effective use of ICT. The means to reach holistic energy management can be listed to be system integration [7], SoA [10] and collecting of KPIs [7, 10]. In [10] ICT methodologies are listed that support implementation of holistic EMS, mentioning CEP, Web Services and SCADA.

In this section approaches were presented that can be used to acquire holistic energy management. Holism requires the understanding of the system in the way of recognizing the relationship between the inputs and outputs of the processes when considering the whole energy domain and its subsystems. Therefore modeling of the behavior and means of measuring the system attributes are required.

2.2 Analytics in Energy Management Systems

On top of the collected and quantified data the EMS applies its analytics. The analytics consist of various case-dependent analytic functions that are used to fulfill the require-

ments of the users of the domain. The results gained from the analytics are used to find the optimal energy consumption performance.

This section concentrates on the algorithms and operations currently used in energy management. No references were found about research of using SOA for management of analytic operations.

2.2.1 Analytic Tools in Energy Management

In EMS the analytic tools are used to refine information from measured data, bringing value to the end user. The need for a series of analytic operations is therefore always generated by the end user, changing the requirements for the tools depending on the use case, environment and user's desires. In manufacturing related energy management the expected outcome of the analytic tools commonly relates to optimization and performance improvements.

The common categories for analytic operations are the following:

- i. visualization**
- ii. prediction**
- iii. decision support**

Examples of the products of analytic tools are different graphs, illustrating the data with suitable graphics, having more demonstrative power than plain data. In prediction statistics are used to recognize trends within the data that allow the future estimations. In decision support different scenarios can be compared with each other, for example when selecting a device to be installed between multiple device types.

Visualization of data brings many benefits. It can be a more powerful way of displaying information than raw data. Also it allows the comparison of different data sets. In [19], *Intelligent Energy Management Platform for Buildings* (INTELLEM) tool is presented, The tool is designed for detailed monitoring and analysis of energy performance in buildings and their subsystems with a visualization layer that has wide capabilities of displaying graphics on base of raw data. The visualization layer is implemented as a Java Script application, utilizing the OpenU15 framework. Figure 2 presents a Sankey Diagram drawn by the INTELLEM's visualization layer, illustrating the energy sources and sinks in a building.

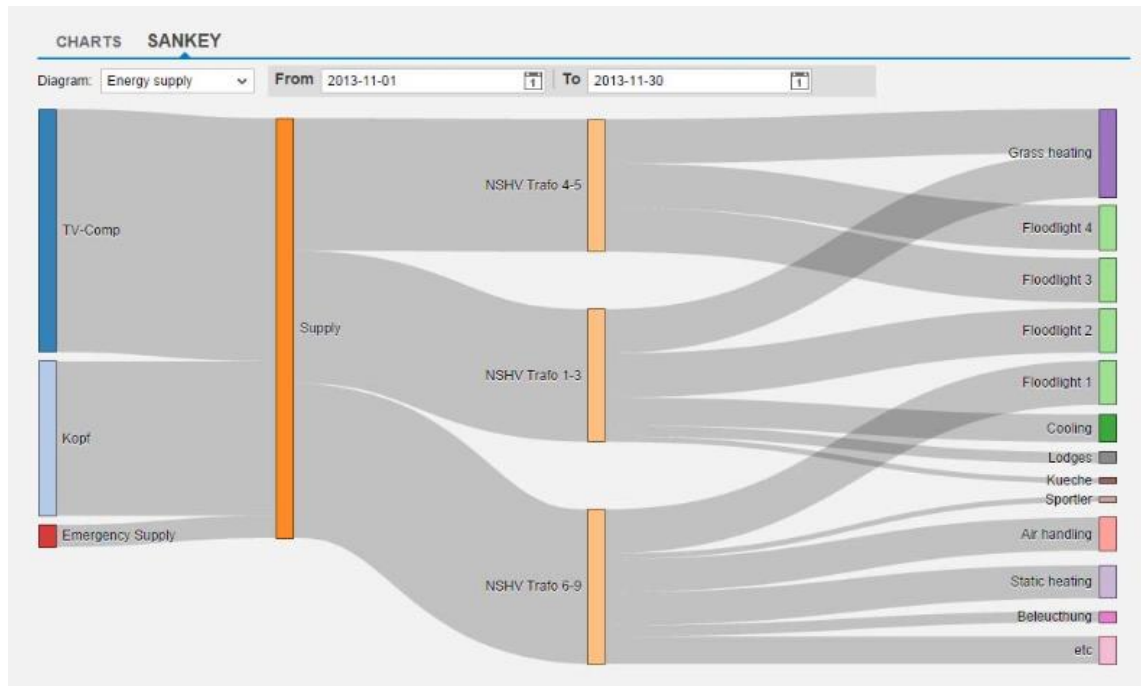


Figure 2: Data visualized into a Sankey diagram. [19]

A prediction algorithm is presented in [20] that uses Support Vector Regression (SVR) method to predict the lighting energy consumption within a building. The algorithm was implemented following the mathematical theory of SVR and applied to real data collected from a building. The results show that the algorithm provides accurate results when estimating the non-linear relation between the lighting energy consumption and its impact factors.

In [21] a decision support system is studied where two clustering algorithms were applied to data consisting of measured values of active power, reactive power and current drained by power sockets. The target was to identify the plugged-in devices and control them according to the status of the overall household system. First data values are attempted to be clustered by using the minimal distance criteria. If the distance comparison indicates no clear clustering condition, clustering by box-dimension criterion is applied. This approach is based on comparing the effect of adding a data value to one of the clusters, and therefore evaluate the cluster to be selected by the effect such positioning. The clustering results are used to identify the operating states of the appliances.

The algorithms used in the research attempt to integrate case-suitable characteristics from other spatial clustering algorithms used in data mining, such as partitioning, hierarchical, density-based and grid-based clustering. The specific characters mentioned in the research are used to embed the complete data set into the potential clustering space and to filter noise with division hierarchical clustering. This illustrates the difficulty of building a suitable analytic tool as they need to be very case specific; organization and reusability of analytic tools are difficult to achieve.

Sophisticated algorithms are being designed for EMS in recent research to provide case-specific means to manage demanding requirements. To enhance the development

of reactive EMS for charging of electric vehicle batteries a real-time algorithm was designed in [22]. To enhance the quality of a microgrid optimization algorithm a set of genetic algorithms were applied in [23] that dynamically optimize the optimization algorithm. For systems that are difficult to model fuzzy algorithms can be applied as was performed in [24] of BAS on behalf of handling the partly random effects of weather conditions.

2.2.2 Review on Data Mining in Manufacturing Systems

Manufacturing facilities create information on different ISA-95 layers. It is expected that emerge of the IoT will rapidly increase the amount of available data. *Data mining* is a methodology that permits the discovery of information and underlying patterns from large data sets by utilizing models built on rules, concepts, patterns, anomalies and trends [25]. In order to operate on large amounts of disperse data the EMS need to utilize data mining characteristics.

Data mining correlates with the concept of *Big Data* [26]. Big Data is a term applied to data sets whose size is beyond the ability of traditional tools to undertake their acquisition, access, analytics or application in a reasonable amount of time. The spread of IoT to the manufacturing industry affects the amount of measured and controllable information, and therefore requires the adaptation of Data Mining concepts to the manufacturing industry.

In [25] the data mining algorithms in manufacturing are divided into six implementation specific categories: *customer relationships, engineering design, manufacturing systems, equipment maintenance, fault detection and quality improvement, and decision support systems*. Added to the variety in categories the algorithms are also affected by the nature of the manufacturing processes itself; time scales, measurement intervals, measurement reliability and accuracy affect the conditions set for the data mining algorithms. Two data mining applications differing in the nature of the manufacturing process are e.g. printed circuit board manufacturing [25] and machining of composite materials [26].

A method of data mining is presented in [27] to be used for building energy management attempting to predict comfortable room technologies. The data mining is implemented by utilizing the decision trees method and knowledge discovery in databases (KDD). KDD provides a method of efficiently preparing the data for further knowledge refining. The method is illustrated in Figure 3: Knowledge Discovery in Databases [28]. Decision trees based classification is used to predict the result based on the measurements and the user comfort.

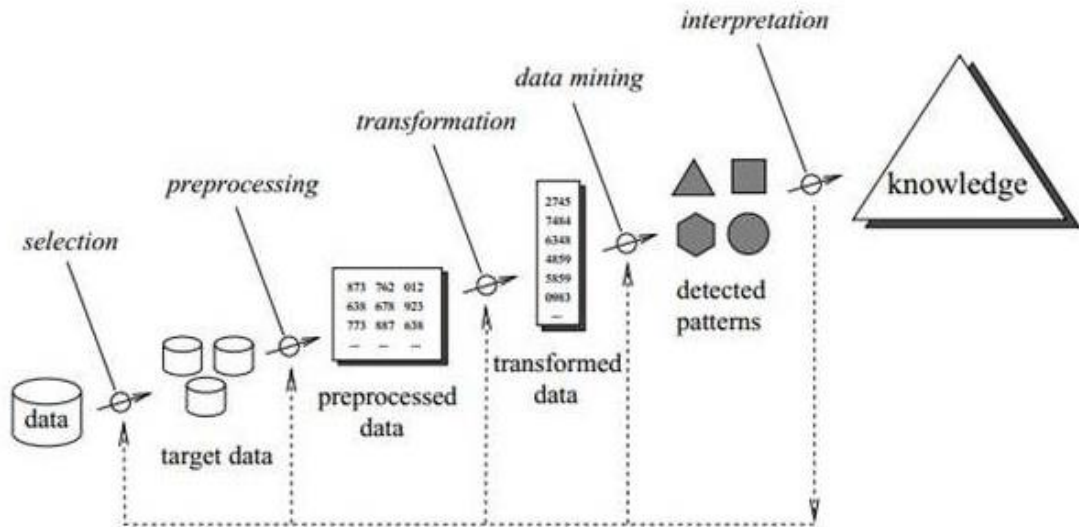


Figure 3: Knowledge Discovery in Databases [28]

In the performed research KDD is the series of methods that transform the raw data measured by wireless sensor networks (WSD) into usable format. The data preparation process includes selection and sampling, cleansing and transformation. Data mining part consists of data analysis and generated output. Post-processing includes visualization and evaluation of mining results. [27]

The transformation of data into KPIs can be seen as an integrated part of the KDD process. There have been attempts to automatize the acquisition of KPIs by using means such as complex event processing. [27]

Data mining can be seen as the complete process of transforming a large set of raw data into usable format and then applying a tuned set of use-case specific algorithms to offer added-value knowledge for the end users. In manufacturing industry it is very difficult to implement any pervasive approach that would provide a solution for various fields and use-cases in manufacturing. This poses a need for sophisticated means of managing the data mining applications.

2.2.3 A Case of Study: URB-Grade

The goal of the URB-Grade project is to design a software-based decision-support tool that is used in the retrofitting of urban area infrastructures. The expected improvements are related to financial, ecological and satisfactory aspects. The solution is based on comprehensive awareness of energy-related behaviors that includes the transformation of data gathered by a WSN into KPIs and user knowledge. [8]

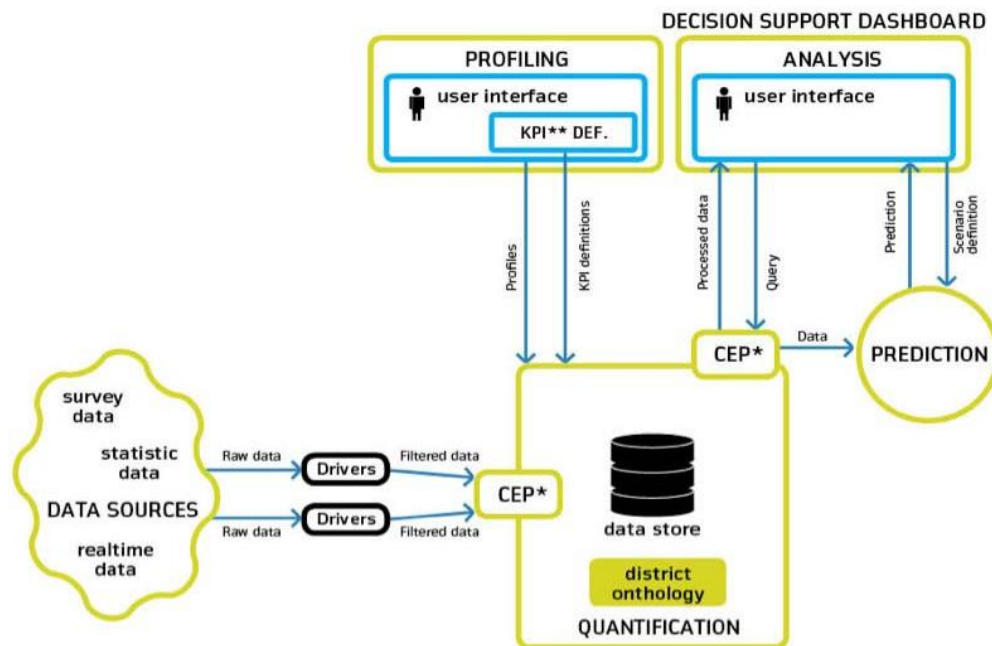
URB-Grade's decision-support tool allows its users to execute various analytical measures on the aggregated data. The results are visualized and presented in a format suitable for the user. Prediction analyses and scenario comparison are included to help the planning of future actions. Concretely this can be e.g. upgrading the bulbs of street lights and optimizing the illumination levels to match people's comfort levels. The deci-

sion-support tool attempts to reveal the existing potential of different investments into the infrastructure. [8]

In order to make the system perform effectively a set of functional and non-functional requirements have been defined by the stakeholders of the URB-Grade project: real-time measurements, automatized analytic processes and combination of multiple data sources [8]. These increase the reactivity of the solution and allow its use in various scenarios in urban areas.

URB-Grade combines modern ICT paradigms such as SOA, EDA, CEP, Event Stream Processing, semantic technologies and cloud computing. Event-based architecture supports the responsiveness and the state-based behavior. SOA makes the system maintainable and adaptable. Cloud computing provides the scalability and semantic technologies support description of relationships and meanings. [8]

The project defines a platform that consists of three modules: “Profiling”, “Quantification” and “Analysis and Forecasting”. Profiling module allows the user to input as-is information of the district into the system. Quantification module uses the defined profiles to aggregate KPIs from the measured data. Analysis module together with Prediction module offer services on top of measured information providing higher-value knowledge, acquired through data fusion and mining. The relationships between system components are presented in Figure 4. [8]



*CEP-Complex Event Processing
** Changing only with administrator role

Figure 4: URB-Grade conceptual architecture. [29]

URB-Grade project introduces multiple resourceful concepts for modern energy management together with modern software technologies. The flexibility provided by

the system architecture makes it possible to apply the solution to urban retrofitting regardless of the measured data.

The project contains many aspects that would be favorable also in the field of manufacturing. Reactiveness and responsiveness are important characteristics in manufacturing processes that normally operate with optimized performance. The holistic approach and the technology stack embracing web technologies and modern integration technologies can be applied in manufacturing use cases.

2.3 Service-Oriented Computing

The target of this thesis is to present an orchestration process of analytic services. Service orchestration is tightly coupled with the concept of SOA and SOC and therefore their state of the art research is reviewed in manufacturing.

This section reviews the state of research and applications that can be applied for managing the orchestration of services.

2.3.1 SOA in the Manufacturing ICT Systems

SOA is a paradigm for organizing and utilizing distributed capabilities that may be under the control of different ownership domains. [30] SOA enables enterprises to build functionality on top of a set of services and therefore enhances reusability, scalability and flexibility. [31] SOA has been supporting the emergence of new innovations in manufacturing as presented next.

In [32] the requirements of modern manufacturing systems are reflected against the SOA principles. The trend in manufacturing is moving from mass customization towards extreme customization where the produced goods that are designed by the customers. It is stated that the ISA 95 architectural model still dominates in the manufacturing realm, but acquiring the future goals requires the adaptation of SOA to the manufacturing enterprise ICT architecture. It is stated that SOA realm is required in the ISA 95 layers in order to make them cooperate seamlessly and to enable the information to flow responsively to and from the neighboring layers. All the enterprise functions, including the manufacturing devices, are managed as services. Paper presents that effective application of SoA has been studied in manufacturing systems containing 10,000 distributed devices. The future research is being targeted on migration, engineering and performance optimization. [32]

Embracing the SOA concept allows the manufacturing enterprises to accustom their operations to the current need set by the markets. The operations are managed by building processes from the service by selecting the ones with the favourable output. Energy management can also be seen as a concept that needs to be integrated into the manufacturing enterprise processes, which makes SOA an architectural requirement for implementing the EMS. SOA offers means to manage the existing complexity. In SOA the service provider of analytic services may as well be the home enterprise, a partner enterprise or any third-party provider. A common example of is the weather services;

companies attempt to concentrate on their key competence and depend on their contractors in subsidiary matters.

2.3.2 Enabling the SOA with Enterprise Service Bus

Enterprise service bus (ESB) is an architectural model that enables the integration of various heterogeneous systems. The ESB provides a common connectivity layer for non-protocol restricted services. An ESB service is a software component that is described with metadata, allowing the services to be of any protocol. The ESB infrastructure provides means to build applications of top of the dynamic service set, perform the required mediational operations (e.g. message transformations and routings) and connect with the required networks. [33] The logical model of SOA Reference Architecture is presented in Figure 5.

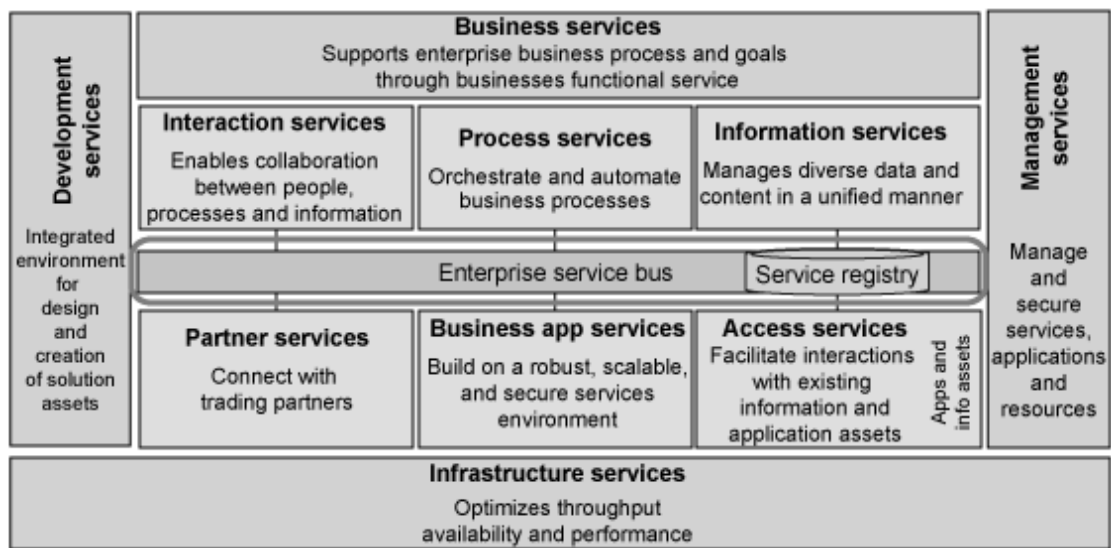


Figure 5: ESB architecture [34]

ESB offers a versatile platform for implementing enterprise software by acting as an integration platform. Therefore ESB supports the implementation of modular applications. ESB has been steadily evolving to be the dominant implementer of SOA.

There are both open-source and proprietary ESB products available. Open source ESBs leverage the open standards and the open source communities for support, which can be seen as strong benefits. Mule ESB [66], Apache ServiceMix [67], WSO2 [68] and Petals ESB [69] are examples of open source ESB products. These products implement the same ESB architecture with different software technologies and their targeted use cases, regarding the heaviness or lightness of the solution.

In PLANTCockpit research projects ESB was applied to gain a loosely-coupled and highly distributable system. The system functionality was implemented as Function Blocks, following the IEC-61499 standard [35]. The approach allows the reuse and configuration of the system components. Function blocks are deployed on the ESB and they encapsulate the communication means with other systems or sources.

In [36] an approach of managing engineering processes on top of ESB based solution is presented. Service components and a business process engine executing com-

posed services are deployed on the ESB. The research applied open source Mule ESB and Activiti BPM tool in implementation. The combination of ESB and BPM allows an effective way to organize the enterprise processes.

Companies have been adopting ESB based solutions as a surging trend. ESB architecture permits a flexible platform for enterprise scale distributed applications. By deploying the EMS on ESB it can be integrated to the existing enterprise applications. ESB acts as a suitable platform for service orchestration and execution of business processes.

3. METHODOLOGY

This section describes the tools and technologies used for the implementation of the analytic service orchestration.

3.1 An Orchestration Process of Analytic Services

This section presents an orchestration process of analytic services that consists of following steps:

- i. Make analytic services discoverable
- ii. Utilize an orchestration engine to build and execute analytic processes
- iii. Define how the interactions between analytic process and analytic services are managed
- iv. Deploy the solution to a suitable platform and make it available for the users
- v. Implement a suitable client to interact with the analytic processes

3.1.1 Conceptual Architecture

The proposed implementation presented in Figure 6 consists of the following software components: Analytic Manager, a number of analytic services, Client and Data Storage. The components use Internet as the communication medium.

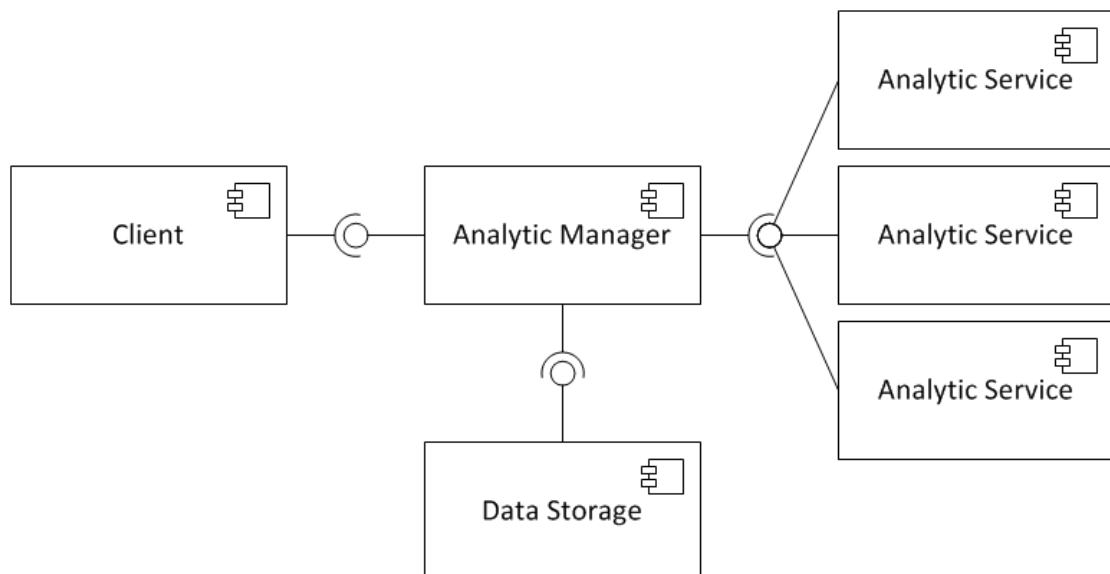


Figure 6: The basic architecture of the proposed solution: Client, Analytic Manager, Data Storage and a number of analytic services.

Analytic Manager is the core component of the system and contains the main functionality required by the analytic service orchestration. Analytic Manager acts as an orchestration engine composing analytic processes from a dynamic set of analytic services. Definitions for analytic service and analytic process are given later in this section. Analytic Manager manages the execution of analytic processes based on requests received from the Client.

In the conceptual architecture Analytic Manager connects to multiple analytic services and utilizes them in order to perform the analytic process orchestration. Analytic Manager publishes the available analytic processes as services to the Client. Analytic Manager connects to the Data Storage in order to acquire data.

Analytic service is defined followingly: analytic services are services that follow the SOA standard and implement the analysis and analytic algorithms required by the EMS. Analytic services are case specific for each use case, which means that the same set of analytic services cannot be applied in each use case. Each algorithmic operation produces new information out from the initial data that provides value to the end user. Analytic services are built into analytic processes to gain analytic knowledge.

And an **analytic process**: an analytic process is an orchestration of a set of analytic services. An analytic process composes a set of analytic services into an executable service that provides the users with the combined results of a set of analytic services. With analytic processes it is possible to build complex processes of analytic services that form information and knowledge from energy-related data and KPIs.

Analytic processes refer to analytic services by using a type; therefore analytic processes are not strictly connected to any specific analytic services. Analytic processes select the analytic services to use by comparing their quality provided to the user. The quality of analytic services is described in the next section.

3.1.2 Quality of Analytic Services

Each analytic service has a type that defines the purpose and the knowledge provided for the user. They may be any amount of analytic services of the same type available, and the Analytic Manager chooses the best option available. The comparison between the analytic services of the same type is done by the quality they provide to the user perspective. This approach is part of the holistic solution as the user's perspective to the domain is included in the analytic process execution.

A textual format was designed to model the quality of analytic services. It can be included into the description element of WSDL files. The quality description adheres to the following format:

$$[P_1[A_{1_1}:V_{1_1}, \dots, A_{1_n}:V_{1_n}]; \dots; P_m[A_{1_m}:V_{1_m}, \dots, A_{n_m}:V_{n_m}]] ; m, n \rightarrow \infty$$

Where P, A and V are defined followingly:

- P = user perspective identifier

- A = attribute identifier
- V = attribute value

The quality description describes the analytic service's quality provided for different user perspectives denoted by $P_1 \dots P_n$. It defines a set of attribute identifier - attribute value pairs that the users can use to estimate the provided quality by relying on specific attribute types.

When a user requests for execution of an analytic process the user's perspective is used to calculate the best available instance of the required analytic service type. The user's perspective defines a set of attributes that the user perspective is interested in, and a weight for each of the attributes:

$$[P[A_1:W_1, \dots, A_n:W_n] ; n \rightarrow \infty$$

Where P, A and W are defined followingly:

- P = user perspective identifier
- A = attribute identifier
- W = attribute weight

Therefore the quality that an analytic service provides for a user perspective is calculated as presented as the sum of the attribute values and attribute weights that have a matching attribute identifier (1):

$$Q = \sum_{n=1}^{\infty} (V_n W_n) \quad (1)$$

Therefore the higher the Q , the better the quality provided to the user perspective by the analytic service.

In the context of this thesis the quality information is not tightly defined. The quality differs by the definitions set by the clients that act as subjects. In the context of manufacturing possible quality attributes relate to the functional and non-functional requirements. The selected approach is suitable for describing the non-functional quality of analytic services.

Non-functional properties are for example the response time, accessibility, compliance, availability, successibility, reputation, cost, reliability etc. [WSS-NFP: Tool for Web Service Selection Based on Non-Functional Properties Using Soft Computing]

Functional properties include input, output, conditional output, precondition, access condition and effect of service. These functional properties can be characterized as the capability of the service.

The users must include the information about their perspectives into the requests that are used to initiate the execution of analytic processes. This information is used to calculate quality values for the analytic services by interpreting their WSDL files and their quality descriptions.

3.2 Service-Oriented Architecture

The main requirement of the orchestration process is that the system must comprise the SOA standard; the organization of the EMS functionality as reusable assets and the ability to build end-to-end business solutions are essential to provide a holistic solution. [37] The modularity provided by SOA allows the system to be integrated into existing systems in a flexible manner. ESB-based architectures greatly enhance the overall interoperability.

In order to enable the distributed use of the solution standard web technologies were taken into use. This section describes how the web technologies are used to implement and access the analytic services of the system.

3.2.1 Web Services

The interoperability in Internet is based on a variety of standards. Web Services is a standard based on HTTP providing services for users via Internet. Web Services use Internet protocols for communication. Therefore Web Services can be used to implement distributed systems over the web.

In [38] it is stated that “A Web service is a software system designed to support interoperable machine-to-machine interaction over a network.” Web services are considered as self-contained, self-describing, modular applications that can be published, located and invoked across the Web. [39]

Web services are based on contracts that are implemented by concrete agents and their functionality is requested by the requester agents. WSDL (Web Service Description Language) is the standard description format for Web Services. WSDL uses XML as a common flexible data exchange format, and applies XML Schema for data typing. The structure of WSDL 1.1 is presented in Figure 7.

Service	Contains a set of system functions that have been exposed to the Web-based protocols.
Port	Defines the address or connection point to a Web service. It is typically represented by a simple HTTP URL string.
Binding	Specifies the interface and defines the SOAP binding style (RPC/Document) and transport (SOAP Protocol). The binding section also defines the operations.
PortType	Defines a Web service, the operations that can be performed, and the messages that are used to perform the operation.
Operation	Defines the SOAP actions and the way the message is encoded, for example, "literal." An operation is like a method or function call in a traditional programming language.
Message	Typically, a message corresponds to an operation.
Types	Describes the data. XML Schema is used (inline or referenced) for this purpose.

Figure 7: WSDL 1.1 structure. [40]

The WSDL 2.0 version was published in 2007 attempting to solve a few shortcomings recognized in version 1.1. [41] In document structure the differences are following: definitions became description, portType became Interface, port became endpoint and message was removed and combined with operation.

As the analytic services are implemented as Web Services their physical location is obsolete from the view of the Analytic Manager component. From the business side of view this enhances the effective utilization of resources as the analytic services can be also provided by partner companies.

The analytic services adhere on the use of a uniform Web Service interface it is natural to follow the contract-first approach in development of new analytic services.

With WSDL the Web services can be published and made available for use. Section 3.1.4 describes some of the main Java libraries and frameworks that utilize WSDL.

3.2.2 SOAP

Simple Object Access Protocol (SOAP is a lightweight protocol intended for exchanging structured information in a distributed environment using the standard HTTP request/response model. [42] Figure 8 presents the structure of SOAP messages.

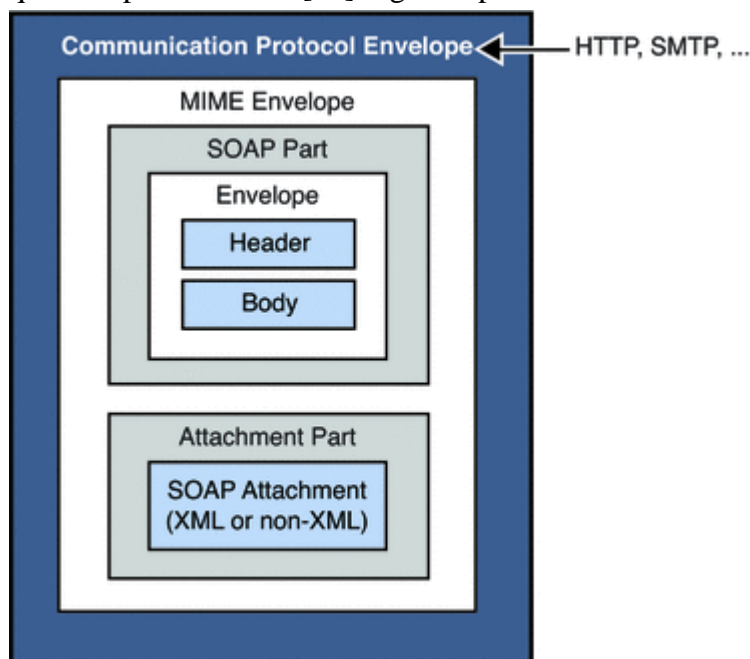


Figure 8: SOAP message structure. [43]

SOAP provides flexible means for applications communicating over the Internet. It grants an extensible messaging framework providing a message construct that can be exchanged over a variety of underlying protocols. [42] SOAP uses platform-neutral XML formats to encode information passed to and returned from remote method calls, which causes some performance reduction as parsing is always required [44].

3.2.3 Java Technologies for Web Services

For Java there are multiple libraries and frameworks that provide interoperability for Java and Web Services

JAXB

XSD are used to define the message schemas used by Web Services. Formal XML messages can be generated by using a XSD file and the contents are both human-readable and machine interpretable. [45].

Java Architecture for XML Binding (JAXB) allows the mapping of Java classes to XML representations and vice versa, by processing the XSD files. Therefore JAXB is a valuable tool when having a Java application utilizing Web Services as the data formats can be easily interchanged in application specific ways.

[45]

JAX-WS

Java API for XML Web Services (JAX-WS) is an API targeted for Web Service-based application development in Java environments. As its core functionality JAX-WS allows the creation of Web Service endpoints and Web Service clients. [46].

JAX-WS enables the development of Web Service interfaces and implementing classes. JAX-WS client may initiate a Web Service from its WSDL URL and call the services via the interface.

3.3 Platform for SOA

In order to meet the integration requirements of the modern EMS and factory systems it was decided to apply ESB as the platform for the solution.

This section describes the selected ESB and its functionality.

3.3.1 OSGi Component Model

OSGi bundles are a common way to deploy application functionality to ESB that act as OSGi frameworks. OSGi (Open Systems Gateway initiative) is a modular framework infrastructure managed by the OSGi Alliance [47]. The goal of the OSGi is to reduce the complexity of modern distributed system development and therefore reduce the development costs. The parts of the OSGi Framework are presented in Figure 9.

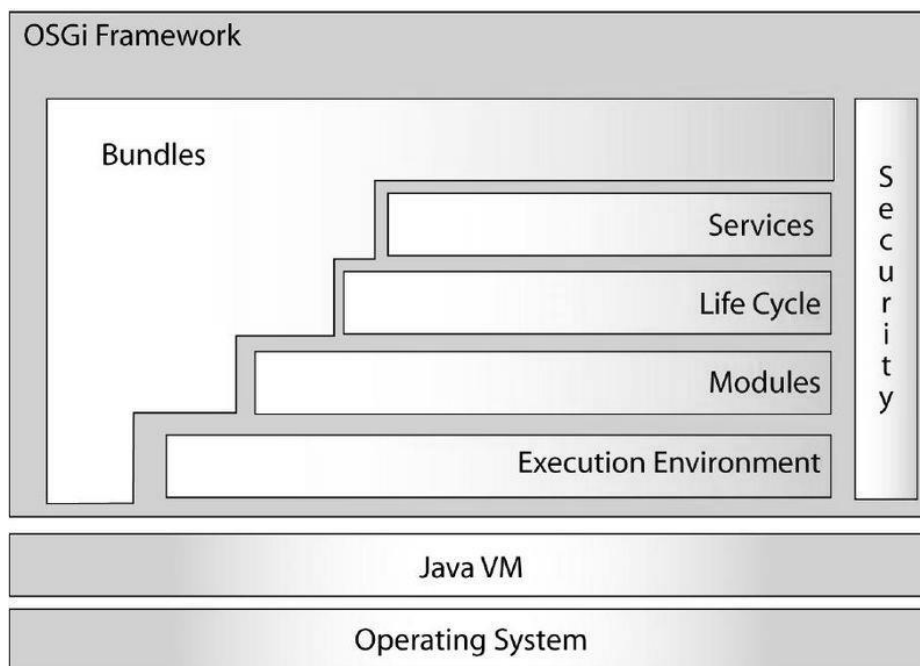


Figure 9: Parts of the OSGi framework. [48]

OSGi specification includes important concepts such as OSGi service registry, OSGi service interface and OSGi bundle lifecycle. OSGi bundles may be dynamically added, deleted, stopped and updated in the ESB application runtime. Their dynamic lifecycle is presented in Figure 10. The approach to use OSGi and ESB also supports the distribution and reusability of the implemented applications. [49]

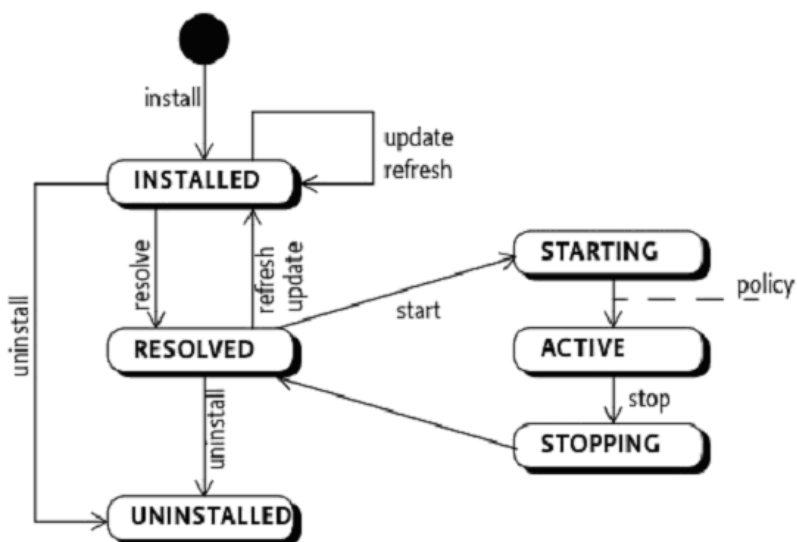


Figure 10: OSGi bundle lifecycle. [50]

The dynamic nature of bundles makes it possible to modify the dependencies between bundles in runtime and therefore removes the need of downtime during software updates.

OSGi's service model describes how bundles cooperate (Figure 11). A bundle can register services, acquire services, and listen for services to appear or disappear. The management of services is managed by *service registry*. [49]

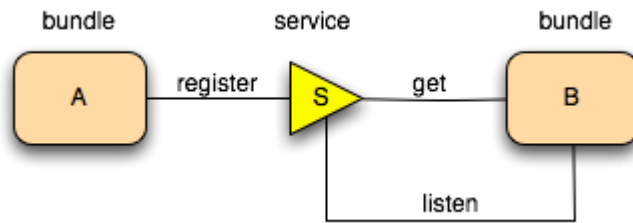


Figure 11: The relationship between OSGi bundles and services. [49]

3.3.2 Apache ServiceMix

Apache ServiceMix [67] is an enterprise-class open-source integration container for building of integrated solutions. ServiceMix embeds a variety of technologies such as ActiveMQ, Camel and CXF assisting the implementation of integrated applications. ServiceMix also embeds Apache Karaf, an OSGi runtime that provides an OSGi container. [51]

Figure 12 presents the architecture of the Apache ServiceMix. The presented technologies are integrated into the container, but new features may also be added application specifically. The target of ServiceMix is to provide an integration platform for development of service-based applications. Therefore ServiceMix can be seen both as middleware and platform for distributed applications. [52]

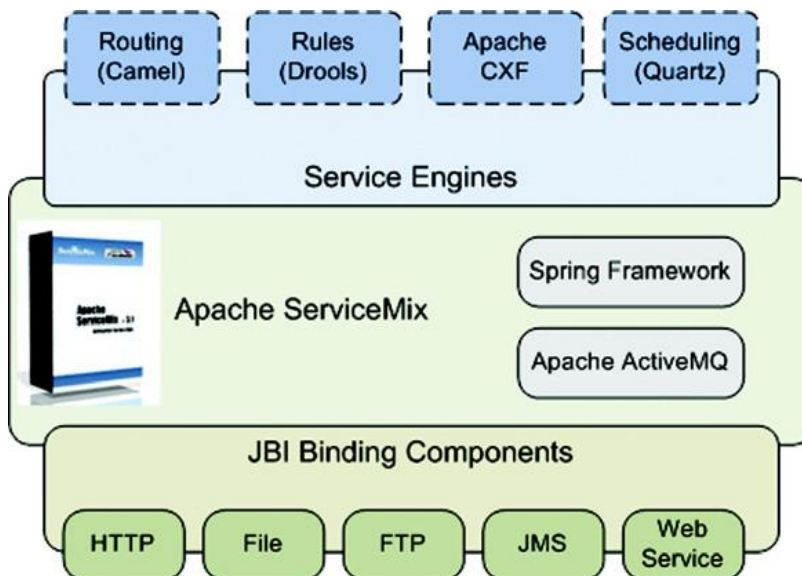


Figure 12: Apache ServiceMix components. [53]

Apache ServiceMix offers means of connecting to different endpoint types including OSGi services and HTTP/SOAP Web Services. In the scope of this thesis ServiceMix is applied as the platform to deploy the solution of the analytic service orchestration. The web technologies described in section 3.1 are supported by ServiceMix. The application functionality can be deployed to ServiceMix wrapped into OSGi bundles.

3.4 Management of Analytic Processes

“*Business Process Management (BPM) includes methods, techniques, and tools to support the design, enactment, management and analysis of operational business processes.*” [54] BPM includes activities such as business process modelling, definition and monitoring.

This section describes how jBPM and Drools can be used together to model and execute business processes modelled in BPMN 2.0. BPM is applied to analytic processes.

3.4.1 BPMN 2.0

Business Process Model and Notation (BPMN) standard provides means for graphically describing business processes. [55] The BPMN 2.0 allows graphical modelling of business processes consisting of states and conditions. The common BPMN elements are the following [55]:

An **Activity** is work that is performed within a business process. An activity can be atomic or non-atomic that triggers an execution of a sub-process. The activities can be for example **user tasks**, where an action is required from a human user. Another type of activities is **service tasks** that trigger the execution of some software service.

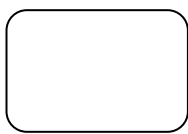


Figure 13: An Activity

Gateways are used to control how sequence flows interact as they converge and diverge within a process. Gateways implement the decision making and branching/joining in the processes.

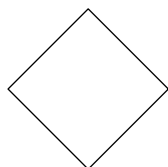


Figure 14: A Gateway

Events are used to signal that something has happened during a process. They affect the process flow and usually have a course or an impact and require or allow a reaction. There are many sorts of events such as start event, end event and intermediate events like signal, message, timer and error events.

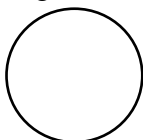


Figure 15: An Event

Transition points the movement between the process states. In order for the transition to fire the transition condition needs to be met. The transition condition may include the state of the process, events and process variables.

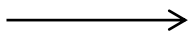


Figure 16: A Transition

Lanes are used to organize and categorize activities within a **Pool**. Pools are used to couple lanes under the same flow of execution.

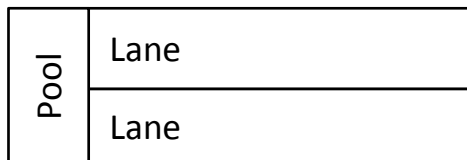


Figure 17: Pool and Lanes

3.4.2 jBPM workflow suite

jBPM open source workflow suite [56] is a tool for designing BPM processes with BPMN 2.0. jBPM includes a process engine that integrates business rules and event processing to execute BPMN processes. jBPM 5.4 was selected to be the tool to implement and execute the analytic processes. jBPM offers the means to perform the service orchestration. jBPM processes can be executed within Apache Karaf by deploying the following OSGi bundles:

```
org.jbpm/jbpm-bpmn2/5.4.0-Final
org.jbpm/jbpm-flow/5.4.0-Final
org.jbpm/jbpm-flow-builder/5.4.0-Final
```

jBPM supports integration with OSGi and Spring Framework, JPA and JTA, and complex event processing via Drools rule engine [57]. Drools is a Business Rules Management (BRM) system that provides jBPM with business rules management support. Drools allows the combining of business rules to the business processes managed by the jBPM. jBPM 5.4 uses Drools version 5.5.0.

The analytic processes are modeled by using the BPMN 2.0 notation using following notions: *service tasks*, *gateways*, *sequence flows*, and *start* and *end events*. Each analytic process may traverse through any kind of path following the BPMN 2.0 specification. [55] [56]

The analytic services are modelled by using *service tasks* of jBPM. Service tasks are used to model automated units of work that should be executed in a process. The implementation of the service task algorithm is performed with dedicated Java classes, which gives a high-degree of freedom for the developer.

jBPM 5.4 has an Eclipse-based editor that allows the graphical creation of business processes. Figure 18 presents basic editor view.

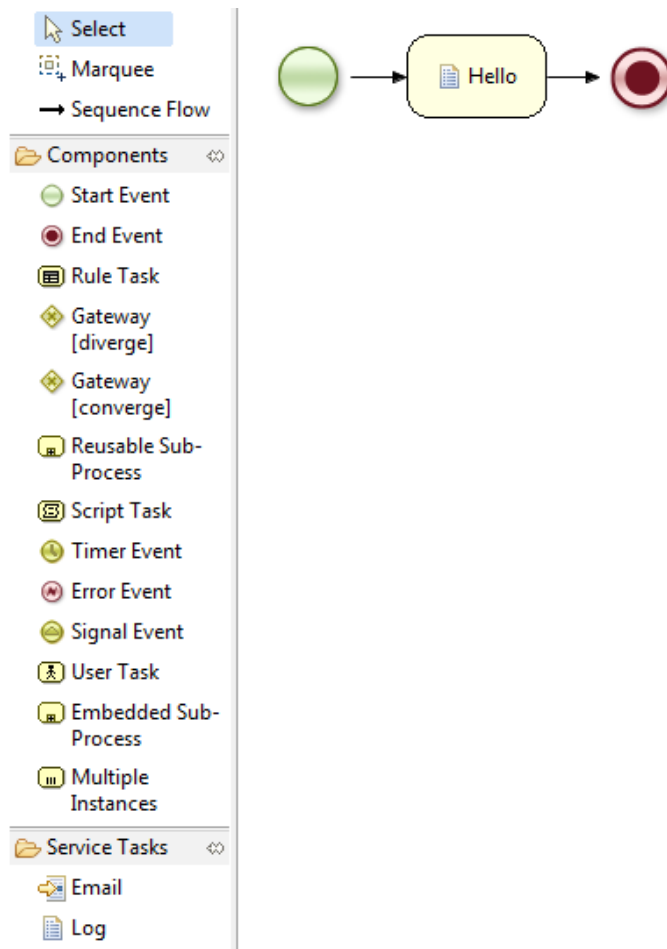


Figure 18: An editor view in jBPM 5.4 editor for Eclipse IDE.

3.5 Data handling in Analytic Processes

The data to be used in analytic service orchestration requires some specific means. This section presents approaches that should allow a generic presentation in order to provide a holistic solution.

3.5.1 E-Nodes

The concept of E-Nodes presented in the Odysseus research project is applied for organization of KPIs. As was stated in [18] “E-Nodes are nodes that produce and/or consume energy of any form”. With E-Nodes it is possible to present both raw data and quantified KPIs in a structured form. In this thesis E-Nodes are used to describe the data that the Analytic Manager required from the Data Storage.

The KPI data processed in the analytic processes is always linked to some entity whose performance the KPI describes. Examples in the manufacturing domain could be entities such as production line, production cell and a robot. Each of them has its own KPIs, included also the possibility of composite dependencies: production cell is a part of the production line, and the robot is part of the production cell. Equivalent structure is presented in Figure 19.

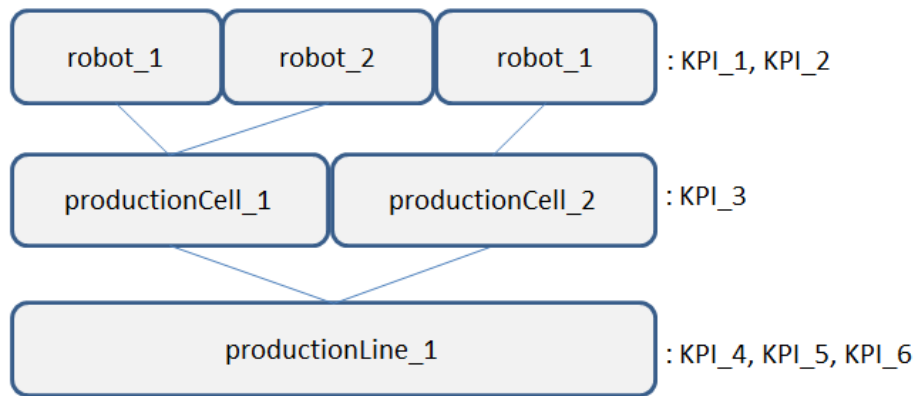


Figure 19: Application of E-Nodes for modeling manufacturing related information.

In this thesis E-Nodes are applied in the data acquisition. E-Nodes are used to describe the structured data required by an analytic process. Listing 1 presents a XML presentation using the entities defined in Figure 19.

```

<tns:eNode id="productionLine_1">
  <tns:data seriesId="series_1" type="float">KPI_4</tns:data>
  <tns:data seriesId="series_2" type="float">KPI_5</tns:data>
  <tns:data seriesId="series_3" type="int">KPI_6</tns:data>
  <tns:eNode id="productionCell_2">
    <tns:data seriesId="series_4" type="string">KPI_3</tns:data>
    <tns:eNode id="robot_1">
      <tns:data seriesId="series_5" type="int">KPI_1</tns:data>
      <tns:data seriesId="series_6" type="int">KPI_2</tns:data>
    </tns:eNode>
  </tns:eNode>
</tns:eNode>

```

Listing 1: Sample XML presenting an E-Node structure.

The definition of the contents the presented in the XML:

- eNode : presents an E-Node entity
 - id : unique identifier
- data : presents data linked to an E-Node. Value defines the target data.
 - seriesId = unique identifier given for the resulting data
 - type = expected type of the data

3.6 Other Software Tools

The remaining software tools are introduced in this section.

3.6.1 Maven

Apache Maven [58] is an open-source tool for managing of build, reporting and documentation in Java-based software projects. Maven is based on the concept of project object model (POM) that may contain various instructions for the tool, including the project dependencies to external software libraries.

Maven provides a solution for the dependency management that is based on using a local Maven repository. The dependencies that are defined in the project object model are downloaded into the local Maven repository and referred from the software project as Maven dependencies. This approach also provides a centralized way of managing the versions of the dependent libraries. [58]

With Maven it's possible to combine multiple software projects into a single build process as with Maven it's possible to define build hierarchies; a hierarchy of software projects can be built by starting the build process on the root project object model.

3.6.2 Spring

Spring Framework [59] is an application development framework that attempts to straight-forward the software development by offering configurable extensions to the code. Spring is extendable by its nature and offers support also for OSGi development. Spring is targeted to address the complexity in enterprise application development. [60]

The core features of Spring are dependency injection, Aspect-Oriented Programming, dynamic modules, Spring web service framework and support for JDBC, JPA and JMS. [59] Spring Framework provides therefore extensive tools for wide-range software development.

Spring enables loosely-coupled applications that have their dependencies listed in configuration files and injected by the framework. [60] This **Inversion of Control (IoC)** supports reusability, maintainability, increases development velocity and simplifies the application structure by decreasing the amount of the code.

In Java code Spring is configured with inline annotations and XML configuration files.

3.6.3 Mojarra JavaServer Faces

Mojarra JavaServer Faces is a MVC (Model-View-Controller) framework that simplifies development of user interfaces (UI) for Java web applications. [61] It implements the Java ServerFaces 2.0 (JSF) standard. Version 2.1.7 of Mojarra JavaServer Faces was used.

JSF applies component-driven UI design model, using XML files called Facelets views.

In this thesis Mojarra JSF was used to develop a web UI to interact with the Analytic Manager. The UI can be deployed to application servers, such as Apache Tomcat.

4. IMPLEMENTATION

This section presents the implementation of the orchestration process of analytic services. The architecture and functionality of the proposed system are introduced.

Java was selected to be the programming language due to its support for interoperability and platform independency. [62]

4.1 System Architecture and Components

The Figure 20 describes the architecture of the solution. The Analytic Manager is implemented as an OSGi bundle and is deployed on Apache ServiceMix. It is the main component of the system implementing the core functionality required for the orchestration of analytic services. The analytic services of the system are either OSGi bundles deployed on the Apache ServiceMix or web applications deployed on Apache Tomcat 7.0 application server [63].

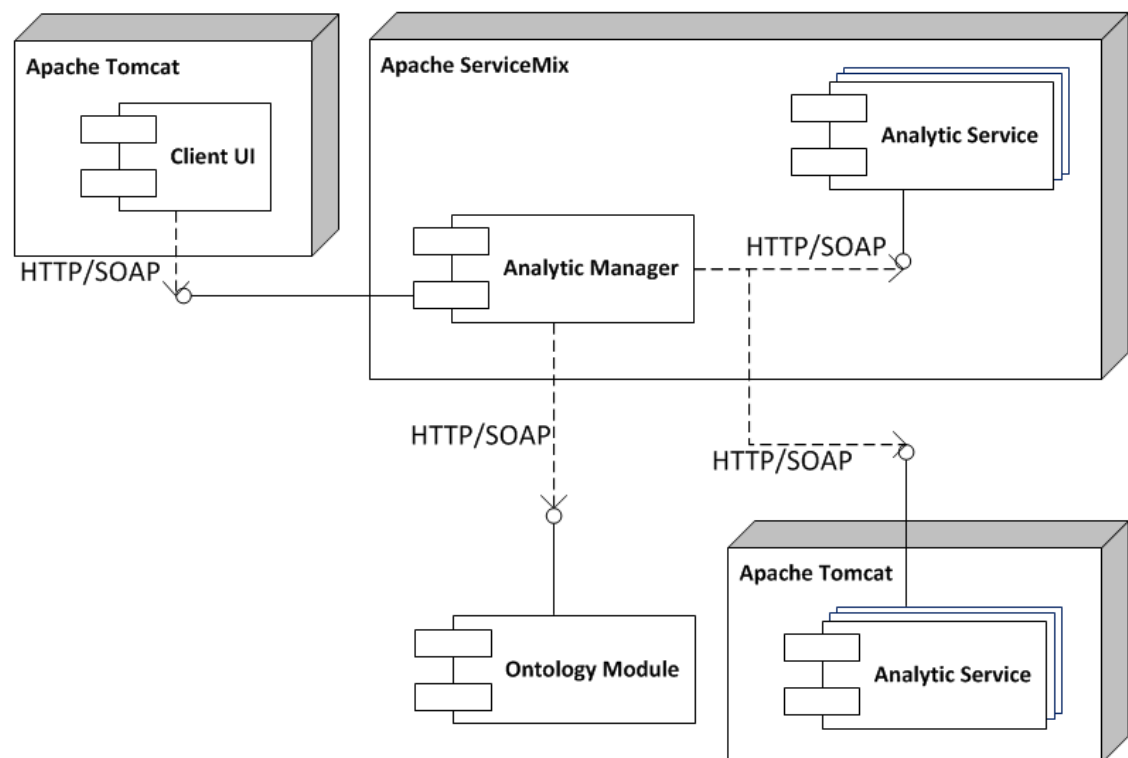


Figure 20: Component diagram describing the architecture of the solution.

Ontology Module is the component responsible for managing the access to the collected energy-related data in the factory environment, acting as the source for the processable data of the analytic processes. Ontology Module does not belong to the scope of this thesis and therefore only its interface is presented.

The analytic services can be deployed to Apache ServiceMix and Apache Tomcat. Deploying to the Apache ServiceMix is the primary method considered in this thesis. However, in this thesis a set of analytic services were utilized that were not deployable to the Apache ServiceMix, which forces the Analytic Manager to use hard-coded URLs in their case.

The Client UI is deployed on another Apache Tomcat instance. All the system components use HTTP/SOAP for intercomponent communication.

4.1.1 Interactions Between System Components

Figure 21 describes the normal flow of interactions between the system components. The interactions between the system components are implemented with HTTP/SOAP Web Services [42]. Apache CXF [64] and Spring Web Services [65] are used as Web Service frameworks. All of the implemented Web Services follow the contract-first approach [66] where the message schemas and WSDLs are defined before implementing any Java classes. This allows the relying on a uniform Web Service interface instead of an application-bound approach.

As presented in the Figure 21 the execution is initiated by the Client UI that sends a request (step 1) to the Analytic Manager containing information of the analytic process, needed data and a set of execution parameters. Analytic Manager acquires the data from the Ontology Module (steps 2–3). Next Analytic Manager begins the execution of the analytic process and makes calls on the analytic services as defined in the selected analytic process (steps 4– $n+1$). During the execution of the analytic process the result is formed and eventually returned to the Client as a response to the initial request (step $n+2$). Table 1 describes the steps presented in Figure 21 with more precision.

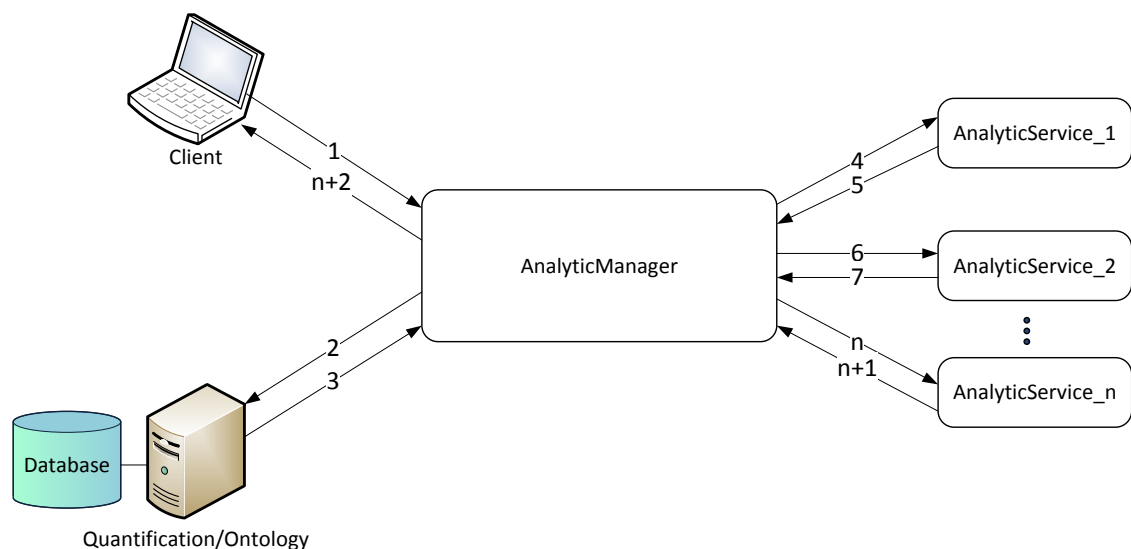


Figure 21: Normal flow of interactions between the system components.

The amount of the analytic services that an analytic process depends on is not limited, nor the order of the analytic services that are executed. Also the presented call to acquire data from the Ontology Module may occur later within the interactions.

Table 1: Sequence of interactions between system components.

No.	Message name	Description
1	analyticProcessRequest	Client sends an initial request containing instructions to execute an analytic process. Analytic Manager validates the request and starts an analytic process in a success case.
2	dataRequest	Within the analytic process a request is sent to acquire data from the Ontology Module regarding the Client's request.
3	dataResponse	Response containing the data is returned.
4	analyticServiceRequest	Request is sent from the analytic process to execute analytic service of type AnalyticService_1.
5	analyticServiceResponse	Response from the AnalyticService_1 is returned.
6	analyticServiceRequest	Request is sent from the analytic process to execute analytic service of type AnalyticService_2.
7	analyticServiceResponse	Response from the AnalyticService_2 is returned.
n	analyticServiceRequest	Request is sent from the analytic process to execute analytic service of type AnalyticService_n.
n+1	analyticServiceResponse	Response from the AnalyticService_n is returned.
n+2	analyticProcessResponse	Result of the analytic process is returned to the Client.

The following sections introduce the system components and their functionality. The Ontology Module component is only introduced at a concept level as it is not in the scope of this thesis.

4.1.2 Analytic Manager OSGi Bundle

This section describes how the Analytic Manager is implemented as an OSGi bundle.

The Analytic Manager component communicates with three other system components. Three Web Service interfaces were defined for Client UI, Ontology module and analytic services in order to interact with the Analytic Manager, each defining a request and a response message type.

Analytic Manager begins an execution of an analytic process when it receives a valid request from the Client UI. The analytic process acquires data from the Ontology Module and executes analytic services regarding the analytic process definition. The flow of sequences is described in the sequence diagram in Figure 22.

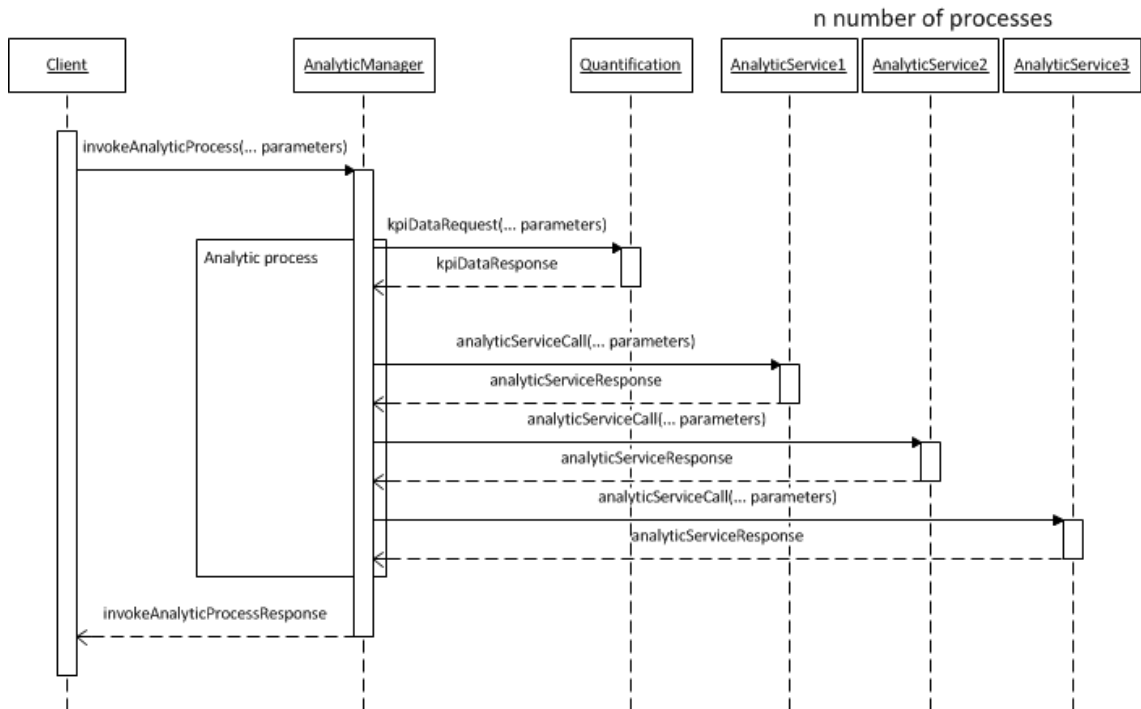


Figure 22: Sequence diagram of analytic process execution.

Within the OSGi bundle the analytic processes are stored as BPMN 2.0 models. Analytic Manager manages the execution of analytic processes by controlling the jBPM 5.4 BPM engine. Each analytic process has its unique name that is used for its invocation. Analytic processes are described with more precision in section 4.2.

Analytic Manager manages the execution of analytic processes with the assist of *rules*. Rules affect the traversing between the process states and the input sent to the analytic services. BPMN models define the possible paths for each instance of analytic process and the rules, the initial input and perspective of the user affect that how the result is eventually formed.

4.1.2.1 Web Service interface

Analytic Manager OSGi bundle publishes a Web Service interface defining two services: *getAvailableProcesses* and *invokeAnalyticProcess*. The clients call these services in order to gain knowledge of the analytic processes provided by the Analytic Manager and then the energy-related knowledge formed by the analytic processes.

getAvailableProcesses service is used to gain information about the currently available analytic processes provided by the Analytic Manager OSGi bundle. Its input message contents are presented in Table 2 and output message contents in Table 3.

Table 2: Input message of the *getAvailableProcesses* Web Service.

Input:	
<i>availableProcessesRequest</i>	
Element:	Description:

<i>availableProcessesRequest</i>	The base element of <i>availableProcessesRequest</i> schema. Contains the <i>requestParameters</i> -element.
<i>requestParameters</i>	Contains a set of metadata-elements setting constraints for the queried analytic processes.
<i>metadata</i>	Key-value pairs defining the query condition.

Table 3: Output message of the `getAvailableProcesses` service.

Output:	
<i>availableProcessesResponse</i>	
Element:	Description:
<i>availableProcessesResponse</i>	The base element of <i>analyticProcessesResponse</i> schema. Contains an unbounded amount of <i>analyticProcess</i> -elements.
<i>analyticProcess</i>	<i>analyticProcess</i> element includes a key-value pair where the key is the unique ID of the analytic process and the value its textual description.

invokeAnalyticProcess is used to invoke the analytic processes. The operation's input message is described in Table 4 and the output message in Table 5. The invocation of this service triggers the execution of an analytic process, which is described in section 4.2.

Table 4: Input message of the `invokeAnalyticProcess` service.

Input:	
<i>analyticProcessRequest</i>	
Element:	Description:
<i>analyticProcessRequest</i>	The base element of the <i>analyticProcessRequest</i> schema.
<i>processParameters</i>	Contains a set of <i>metadata</i> elements including execution parameters for the analytic process. This data includes e.g. the name of the analytic process to execute and the time window of the data series.
<i>metadata</i>	<i>Metadata</i> contains key-value pairs assigning analytic process execution instructions for the Analytic Manager.
<i>eNode</i>	<i>eNode</i> is an recursive elements that is used to form hierarchical structures that explicitly define the data to be processed in analytic processes.
<i>dataId</i>	Sets two attributes for the <i>eNode</i> : <i>type</i> and <i>seriesId</i> .

Table 5: Output message of the `invokeAnalyticProcess` service.

Output:

analyticProcessResponse	
Element:	Description:
<i>analyticProcessResponse</i>	The base element of <i>analyticProcessResponse</i> schema. Contains the <i>processResults</i> -element and a set of <i>outputDataSeries</i> -elements.
<i>processResults</i>	Contains a set of <i>metadata</i> that describes the abstract results of the analytic process.
<i>metadata</i>	<i>Metadata</i> contains key-value pairs describing the result conditions of the executed analytic process.
<i>outputDataSeries</i>	<i>outputDataSeries</i> contains a set of <i>dataValues</i> -elements bound to <i>metadata</i> -elements, describing the numeric outcome of the analytic process.
<i>dataValue</i>	<i>dataValue</i> binds a concrete data value to a timestamp.

4.1.3 Analytic Service OSGi Bundles

This section describes how the analytic services are implemented as OSGi bundles.

The set of analytic services is dynamic; new analytic services may become available or unavailable whenever the ESB is running. Also it is up to the system designer to decide that how the analytic processes are built from the analytic services; analytic processes are case and application specific, but also their reuse is possible.

4.1.3.1 Web Service interface

The Analytic Services share a uniform WSDL interface that defines uniform formats for input and output messages. This allows the analytic services to be organized into sequences that form the analytic processes. Each analytic service utilizes the input and output messages defined in Table 6 and Table 7.

Table 6: Input message of an analytic service Web Service.

Input:	
Element:	Description:
<i>analyticServiceRequest</i>	
<i>analyticServiceRequest</i>	The base element of the <i>analyticServiceRequest</i> schema.
<i>serviceParameters</i>	Contains a set of <i>metadata</i> elements including execution parameters for the analytic service. This data includes e.g. algorithm specific variables.
<i>inputDataSeries</i>	<i>inputDataSeries</i> contains a set of <i>dataValue</i> -elements bound to <i>metadata</i> -elements, describing

<i>metadata</i>	the data to be processed in the analytic service. <i>Metadata</i> contains key-value pairs assigning analytic service execution instructions both generally and <i>inputDataSeries</i> specifically.
<i>dataValue</i>	<i>dataValue</i> binds a concrete data value to a timestamp.

Table 7: Output message of an analytic service Web Service.

Output: <i>analyticServiceResponse</i>	
Element: <i>analyticServiceResponse</i>	Description: The base element of the <i>analyticServiceResponse</i> schema.
<i>serviceResults</i>	Contains a set of <i>metadata</i> elements including variables describing the results of the analytic service.
<i>outputDataSeries</i>	<i>outputDataSeries</i> contains a set of <i>dataValue</i> -elements bound to <i>metadata</i> -elements, containing the concrete results of the analytic service. Each <i>outputDataSeries</i> is mapped to the corresponding <i>inputDataSeries</i> with <i>seriesId</i> parameter.
<i>metadata</i>	<i>Metadata</i> contains key-value pairs assigning analytic service execution instructions both generally and <i>inputDataSeries</i> specifically.
<i>dataValue</i>	<i>dataValue</i> binds a concrete data value to a timestamp.

4.1.3.2 Registration of Analytic Services

The endpoints of the analytic services are registered into Apache ServiceMix's Service Registry that maintains information of the registered services. Services can be dynamically registered and unregistered to the Service Registry following the common OSGi lifecycle as was presented in section 3.2.2.

Analytic Services that are run on the ServiceMix are registered to its Service Registry via Spring Framework. Listing 2 exemplifies Analytic Service's JAX-WS endpoint registration. The attributes of the *endpoint*-element configure the analytic service to start as a Web Service.

```

<?xml version="1.0" encoding="UTF-8"?>
<beans xmlns="http://www.springframework.org/schema/beans"
  xmlns:xsi="
    http://www.w3.org/2001/XMLSchema-instance"
  xmlns:jaxws="
    http://cxf.apache.org/jaxws"
  xsi:schemaLocation="
    http://www.springframework.org/schema/beans
    http://www.springframework.org/schema/beans/spring-beans.xsd
    http://cxf.apache.org/jaxws http://cxf.apache.org/schemas/jaxws.xsd">

  <jaxws:endpoint xmlns:tns="http://analyticsservices.tut.fi/sampleservice"
    id="sampleService"
    implementor="fi.tut.analyticsservices.sampleservice.SampleServiceImpl"
    wsdlLocation="classpath:META-INF/wsdl/sampleservice.wsdl"
    endpointName="tns:SampleService"
    serviceName="tns:SampleService"
    address="http://localhost:8190/cxf/AnalyticServiceSampleService_1">
    <jaxws:features>
      <bean class="org.apache.cxf.feature.LoggingFeature" />
    </jaxws:features>
  </jaxws:endpoint>

</beans>

```

Listing 2: Analytic service registration to OSGi service registry.

In this thesis it is set as a constraint that the information of the analytic services must be accessible from the Analytic Manager OSGi bundle. Therefore the Analytic Manager may use the services of the Apache ServiceMix to discover the available analytic services. The service information is maintained up-to-date with the help of OSGi lifecycle management. Via the Service Registry the Analytic Manager may gain access to the URL of the WSDL file of the analytic service and create a client for it by using the JAX-WS API.

4.1.4 Ontology Module Web Service

Ontology Module provides the Analytic Manager with energy-related data that may be used in the analytic processes. The Ontology Module connects to the databases where the factory's energy-related data gathered from various sources is stored. The data may be either measured raw data or quantified data represented by key performance indicators. Ontology Module also manages the mapping of perspectives to the persisted data enhancing the use of data in a way natural to the different user types. Each user may possess an own perspective on the energy-related data, comprising of lexicon and data models.

Conceptually Ontology Module is similar with the Quantification Module of the URB-Grade project. Quantification Module was described in section 3.2.2.

4.1.4.1 Web Service interface

A WS interface was designed for the Ontology Module. The *dataService* takes a *dataRequest* message as an input and returns *dataResponse* message as an output. In the *dataRequest* message the concept of E-Nodes (section 3.5.1) is applied. The input message is described in Table 8 and the output message in Table 9.

Table 8: Definition of the dataRequest message type.

Input: <i>dataRequest</i>	
Element: <i>dataRequest</i>	Description: The base element of the <i>dataRequest</i> schema.
<i>requestParameters</i>	Contains a set of <i>metadata</i> elements including query parameters for the quantification service.
<i>eNode</i>	<i>eNode</i> elements are used to build trees of recursive <i>eNodes</i> to present hierarchical data structures. Each <i>eNode</i> may have a series of <i>eNode</i> and <i>dataId</i> elements. Each <i>eNode</i> has an <i>id</i> attribute that defines the <i>eNode</i> explicitly in the managed <i>eNode</i> database.
<i>data</i>	<i>data</i> element has a String value that defines the requested KPI type for the specific <i>eNode</i> . <i>dataId</i> has two attributes <i>type</i> and <i>seriesId</i> ; <i>type</i> is the preferred data type to be used for the KPI and <i>seriesId</i> is an ID that is attached to the returned KPI data series.
<i>metadata</i>	<i>Metadata</i> contains key-value pairs that are used to configure analytic service execution.

Table 9: Definition of the dataResponse message type.

Input: <i>dataResponse</i>	
Element: <i>dataResponse</i>	Description: The base element of the <i>dataResponse</i> schema. Contains a single <i>responseParameters</i> element and an unbounded amount of <i>responseDataSeries</i> elements.
<i>responseParameters</i>	Contains a set of <i>metadata</i> elements including information of the query results.
<i>responseDataSeries</i>	<i>responseDataSeries</i> contains an unbounded amount of <i>metadata</i> and <i>dataValue</i> elements. Each <i>responseDataSeries</i> has a <i>seriesId</i> attribute that links the <i>responseDataSeries</i> to the corresponding <i>eNode</i>

metadata

element used in the *dataRequest* message.

Each *Metadata* elements contains a key-value pair that allows the including of additional information.

dataValue

dataValue element has the KPI value as its value. The *timestamp* attribute binds the KPI value to a time value.

4.1.5 Client UI

Client UI was implemented by using the Mojarra JavaServer Faces framework. The framework was described in section 3.6.3. The client UI allows the user to fill in the following fields:

- URL of the Analytic Manager's WSDL
- Amount of parallel threads
- *analyticProcessRequest* message
- *availableProcessesRequest* message
- A button to trigger *invokeAnalyticProcess* service
- A button to trigger *getAvailableProcesses* service
- A button to refresh the responses
- A button to reset the request messages
- A field to displaying received *analyticProcessResponse* messages
- A field to displaying received *availableProcessesResponse* messages

Figure 23 displays the implemented Client UI.

Client UI for analytic manager interactions

Service address

Select thread amount

Analytic process request

```
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<analyticProcessRequest xmlns="http://analytics.tut.fi/analyticManager/executionRequest">
  <!-- only one requestParameters elements -->
  <requestParameters>
    <!-- Unbounded amount of metadata elements -->
    <metadata key='processName'>fi.tut.bpmn.stubTest1</metadata>
    <metadata key='perspective'>perspective1</metadata>
    <metadata key='qualityAttributes'>perspective1[price:0.9]</metadata>
    <metadata key='userDecision'>ParetoChart</metadata>
    <metadata key='startTime'>10000000</metadata>
    <metadata key='endTime'>20000000</metadata>
    <metadata key='userSelection'>ParetoChart</metadata>
    <metadata key='resolutionUnit'>hour</metadata>
    <metadata key='resolutionValue'>12</metadata>
    <metadata key='GenerateDataFillGaps'>>false</metadata>

    <!-- Clustering -->
    <metadata key='legendLocation'>BestOutside</metadata>
    <metadata key='pointStyle'>dot</metadata>
    <metadata key='replicates'>5</metadata>
    <metadata key='numberOfClusters'>2</metadata>
  </requestParameters>
</analyticProcessRequest>
```

Available processes request

```
<?xml version="1.0" encoding="UTF-8"?>
<availableProcessesRequest xmlns="http://analytics.tut.fi/analyticManager/processesRequest">
  <!-- only one requestParameters elements -->
  <requestParameters>
    <!-- Unbounded amount of metadata elements -->
    <metadata key='param1'>val1</metadata>
    <metadata key='param2'>val2</metadata>
    <metadata key='param3'>val3</metadata>
  </requestParameters>
</availableProcessesRequest>
```

Invoke analytic process

Get processes

Show process result

Reset

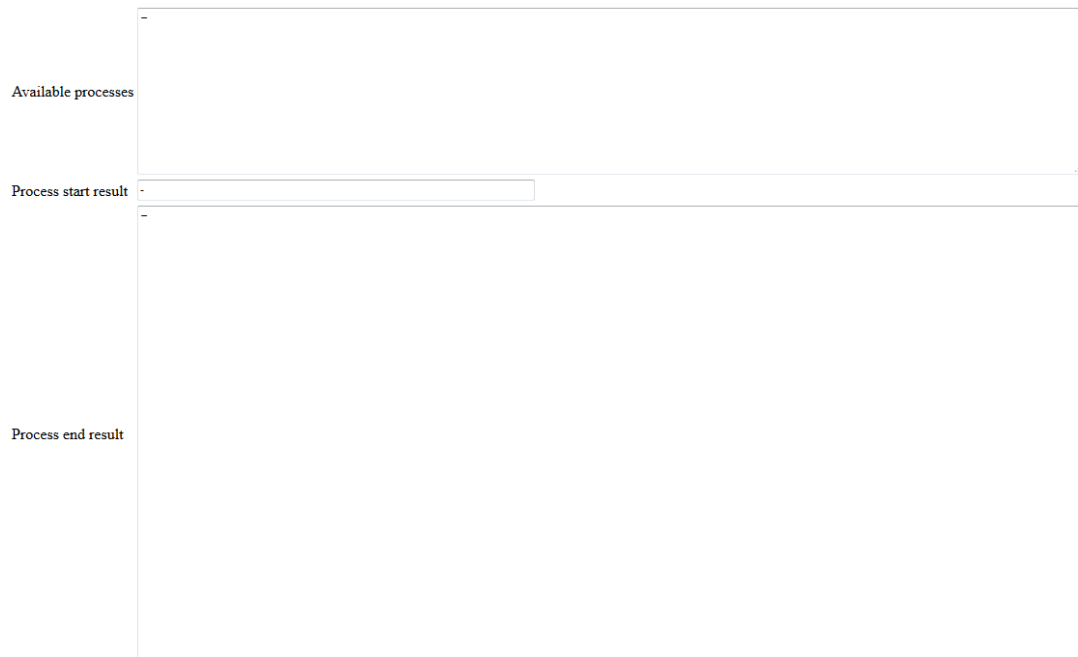


Figure 23: The Client UI

For testing purposes the UI can be set to send multiple requests in parallel threads. This allows the testing of the parallel processing capabilities of the Analytic Manager.

When the Client UI receives an `analyticProcessResponse` message, it checks whether the message contains image data or not. If the received response contains graphics, the contents are displayed in a separate window as presented in Listing 3.

```

Metadata metadata = it.next();
if (metadata.getKey().equals(imageMetadataKey)) {
    sun.misc.BASE64Decoder decoder = new sun.misc.BASE64Decoder();
    byte[] imageDataBytes = decoder.decodeBuffer(metadata.getValue());

    ImageIcon imageIcon = new ImageIcon(imageDataBytes);
    JLabel label = new JLabel(imageIcon);
    JFrame imageFrame = new JFrame();
    imageFrame.setTitle("Analytic process image result");
    imageFrame.setSize(imageIcon.getIconWidth(), imageIcon.getIconHeight());
    imageFrame.setContentPane(label);
    imageFrame.setVisible(true);
}

```

Listing 3: Parsing image result from the `analyticProcessResponse`'s metadata element.

4.2 Analytic Service Orchestration

This section describes the orchestration process of analytic services into analytic processes and their execution.

4.2.1 Injection of Analytic Process Definitions

A Drools work item is needed for each jBPM service task. They are defined in configuration files as described in Listing 4.

```

import org.drools.process.core.datatype.impl.type.FloatDataType;
import org.drools.process.core.datatype.impl.type.ObjectDataType;
import org.drools.process.core.datatype.impl.type.StringDataType;
[
  // the RadarChart work item
  [
    "name" : "RadarChartItem",
    "parameters" : [],
    "displayName" : "RadarChart",
    "icon" : "icons/radarChart.gif"
  ],
  ...
]

```

Listing 4: AnalyticDefinitions.conf defining a Drools work item for RadarChart Analytic Service.

Then the work item definitions are included to drool.rulebase.conf that lists the configurations that are injected into Drools container (Listing 5). This makes the domain node types visible in jBPM process editor (Figure 24).

```
drools.workDefinitions = AnalyticDefinitions.conf
```

Listing 5: Contents of drools.rulebase.conf.

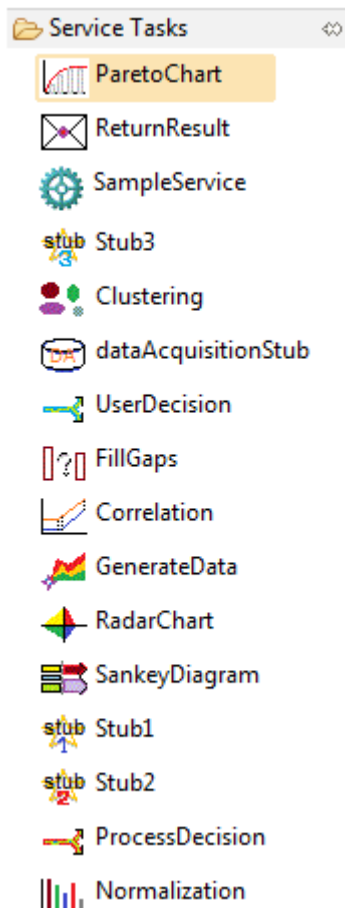


Figure 24: Domain node types injected to Drools.

Each analytic service specific service task shall have a Java class implementing the `org.drools.runtime.process.WorkItemHandler` interface (Listing 6).


```

public class RadarChartHandler extends AnalyticServiceHandler implements
WorkItemHandler {
    @Override
    public void abortWorkItem(WorkItem arg0, WorkItemManager arg1) {
        ...
    }
    @Override
    public void executeWorkItem(WorkItem i, WorkItemManager m) {
        ...
    }
}

```

Listing 6: A `WorkItemHandler` was defined for each service task.

Spring is used to configure the BPMN 2.0 processes and the `WorkItemHandler` implementations required by the service tasks triggering the analytic services. The locations of the BPMN 2.0 models of the defined analytic processes are injected to `AnalyticManager` to instantiate an `org.drools.KnowledgeBase` object (Listing 7). The initiated Knowledge Base is then injected with the definitions of the `WorkItemHandlers` to initiate an `org.drools.runtime.StatefulKnowledgeSession` that is eventually used to create instance of analytic processes and to start them by using the process name defined in the `analyticProcessRequest` message sent by the user. An example set of analytic processes is demonstrated in section 5 (Results).

```

<bean id="kbaseBean" class="fi.tut.processengine.AnalyticProcessEngine"
factory-method="createKnowledgeBase">
<constructor-arg name="processDefinitionsList">
<list>
<value>analyticProcesses/clustering.bpmn</value>
<value>analyticProcesses/correlation.bpmn</value>
<value>analyticProcesses/paretoChart.bpmn</value>
<value>analyticProcesses/radarChart.bpmn</value>
...
</list>
</constructor-arg>
</bean>

<bean id="ksessionBean" class="fi.tut.processengine.AnalyticProcessEngine"
factory-method="createKnowledgeSession">
<constructor-arg name="kbase" ref="kbaseBean" />
<constructor-arg name="eventHandlerList">
<list>
<value>Clustering</value>
<value>Correlation</value>
<value>ParetoChart</value>
<value>RadarChart</value>
...
</list>
</constructor-arg>
</bean>

```

Listing 7: Injecting Drools knowledge base and knowledge session with Spring.

A design decision was made to implement the service tasks in four categories: *Data Acquisition* service tasks are used to import energy-related data into the analytic process via the interface defined in 4.1.4.1, *Analytic Service* service tasks execute a service call on the corresponding analytic service by using the analytic data, *Decision Tasks* service tasks are used to make a decision between possible transitions within analytic process and *Result handling* service tasks are used to export the analytic data from the analytic process to the Analytic Manager.

The implemented service tasks have uniform interfaces, i.e. they all receive and output analytic data, which allows them to be ordered in any order. The designer's responsibility is to design the analytic processes in a reasonable manner.

4.2.2 The Use of the OSGi Service Registry

The OSGi Service Registry of Apache ServiceMix is used to gain information of the available analytic services during analytic process execution.

Analytic Manager OSGi bundle acquires a reference to the `org.apache.cxf.transport.http.DestinationRegistry` managed by the ServiceMix to gain access to the information of the registered analytic services. This Destination Registry contains the destination paths of the services, i.e. the URLs of the WSDL files of the analytic services. The service references are acquired as described in the code snippet presented in Listing 8.

```

ServiceReference[] references =
    bundleContext.getServiceReferences(
        "org.apache.cxf.transport.http.DestinationRegistry", null);
for (ServiceReference reference : references) {
    if (reference != null) {
        DestinationRegistry destinationRegistry =
            (DestinationRegistry) bundleContext.getService(reference);
        Set<String> destinationPaths =
            destinationRegistry.getDestinationsPaths();
        for (String destinationPath : destinationPaths) {
            if (destinationPath.contains("AnalyticService")) {
                internalServices.add(ss);
            }
        }
    }
}

```

Listing 8: Acquiring service references from the OSGi service registry.

From the acquired `destinationPaths` the analytic services are filtered by referring to the naming convention that each analytic service shall have the String “AnalyticService” included in its destination path together with String defining the exact type of the analytic service.

The `destinationPaths` is used later on in the analytic process execution to deduce if the required analytic services are available to execute a requested analytic process. Also the service selection requires handles of the analytic services of the same type as described in section 3.1.2.

4.2.3 Invocation of Analytic Services

When an analytic service needs to be called from an analytic process, a common routine is followed. First the available analytic services of the required type need to be acquired from the Service Registry as described in the previous section.

In the case that there are multiple analytic services of the same type available their WSDL files need to be interpreted to define the quality value they provide for the user perspective.

When one of the analytic process's service tasks is triggered, meaning that its implementer of `org.drools.runtime.process.WorkItemHandler` is called, a series of processing takes place.

Once the URLs to the available analytic services are acquired they are used to read out the WSDL files (Listing 9).

```
public String getQualityParameters(String wsdlUrl) throws Exception {
    WSDLFactory factory;
    try {
        factory = WSDLFactory.newInstance();
        WSDLReader reader = factory.newWSDLReader();

        Definition wsdlInstance = reader.readWSDL(null, wsdlUrl);

        String wsdlQualityDescription = null;
        wsdlQualityDescription =
            wsdlInstance.getDocumentationElement().getTextContent();
        wsdlQualityDescription = wsdlQualityDescription.substring(
            wsdlQualityDescription.indexOf("[") + 1,
            wsdlQualityDescription.lastIndexOf("]"));

        return wsdlQualityDescription;
    }
}
```

Listing 9: Reading the quality description from analytic service's WSDL.

The quality information is extracted from the WSDL file and injected into a dedicated data container (Listing 10).

```

public QualityAttributeStorage(String wsdlUrl, String qualityDescription)
{
    analyticServiceWsdlUrl = wsdlUrl;
    perspectiveMap = new HashMap<String, Map<String, Double>>();

    String[] perspectiveStrings = qualityDescription.split(";");

    for (String perspectiveString : perspectiveStrings) {
        String perspective =
            perspectiveString.substring(0, perspectiveString.indexOf("["));
        String dataArea =
            perspectiveString.substring(perspectiveString.indexOf("[")+1,
                perspectiveString.lastIndexOf("]"));

        String[] qualityStrings = dataArea.split(",");

        Map<String, Double> qualityMap = new HashMap<String, Double>();

        for (String qualityString : qualityStrings) {
            String[] dataStrings = qualityString.split(":");
            String attribute = dataStrings[0];
            Double value = Double.parseDouble(dataStrings[1]);
            qualityMap.put(attribute, value);
        }
        perspectiveMap.put(perspective, qualityMap);
    }
}

```

Listing 10: Obtaining the quality information from quality description.

Then the quality attributes and the set weights are used to calculate the quality value for each analytic service (Listing 11). The formula for the quality value calculation is presented in section 4.2.7.

```

Iterator<Entry<String, Double>> it =
    (Iterator<Entry<String, Double>>) userConstraints.entrySet().iterator();
while (it.hasNext()) {
    Entry<String, Double> entry = it.next();
    Double value = perspectiveSpace.get(entry.getKey());
    double weight = entry.getValue();

    qualityValue += value * weight;
}

if (qualityValue > bestQualityValue) {
    bestQualityWsdl = qAS.getAnalyticServiceWsdlUrl();
    bestQualityValue = qualityValue;
}

```

Listing 11: Calculation of the quality value.

Once the analytic service that provides the Client with most value has been selected a Web Service client is created by using JAX-WS and the analytic service is invoked with the analytic data. The result is then aggregated into the analytic data managed by the analytic process (Listing 12).

```

AnalyticServiceCommunication aSC = AnalyticServiceCommunication
    .getAnalyticServiceCommunication();
AnalyticServiceRequest request = new AnalyticServiceRequest();
request.getInputDataSeries().addAll(
    dataContainerInput.extractInputData());
request.setServiceParameters(new ServiceParameters());
request.getServiceParameters().getMetadata()
    .addAll(dataContainerInput.extractMetaData());

String analyticRequestMessage = aSC.convertToXml(request,
    AnalyticServiceRequest.class);
analyticRequestMessage = aSC.replaceNamespace(analyticRequestMessage,
    analyticServiceNamespace);

AnalyticServiceResponse responseResult = new AnalyticServiceResponse();

responseResult = (AnalyticServiceResponse) aSC.dispatchCall(
    analyticServiceWSDL, SERVICE_NAME, PORT_NAME,
    analyticRequestMessage, AnalyticServiceResponse.class);

```

Listing 12: Invocation of the selected analytic service.

The result is then aggregated into the analytic data managed by the analytic process (Listing 13).

```

DataContainer dataContainerOutput = new DataContainer();

dataContainerOutput.setProcessParameters(dataContainerInput
    .getProcessParameters());
dataContainerOutput.mergeWithOutputMetadata(
    responseResult.getServiceResults());
dataContainerOutput.mergeWithOutputData(
    responseResult.getOutputDataSeries(), false);
dataContainerOutput.mergeWithInputMetadata(
    dataContainerInput.getDataSeriesList());

Map<String, Object> results = new HashMap<String, Object>();
results.put("dataContainerOutput", dataContainerOutput);

m.completeWorkItem(i.getId(), results);

```

Listing 13: Merging the result to the analytic data.

4.2.4 Data structure for Analytic Data

In order to manage the analytic data that is processed within the analytic processes a suitable data structure “DataContainer” was designed. This data structure allows the management and combining of the results gained from the analytic services. DataContainer is included as a parameter in each analytic process and it is passed to the WorkItemHandlers of service tasks.

First the class contains a Map<String, String> processParameters that is initialized from the initial *analyticProcessRequest*. List<DataSeries> dataSeriesList contains objects that contain the concrete data series, series ID and metadata. The outline of DataContainer is presented in Figure 25.

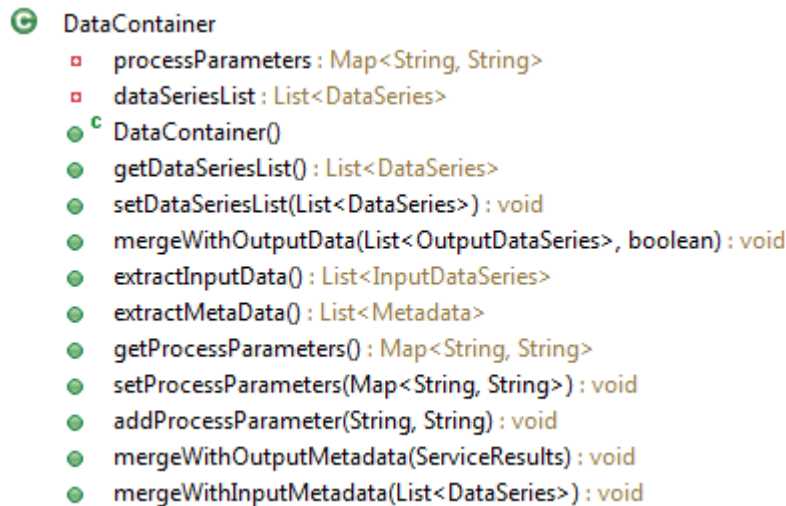


Figure 25: Outline of DataContainer class.

Within analytic processes a DataContainer instance is included as a process parameter (Figure 26), which is then utilized in analytic service service tasks that use it to copy the input parameter and the result. Figure 27 presents the parameters used for analytic service service tasks.

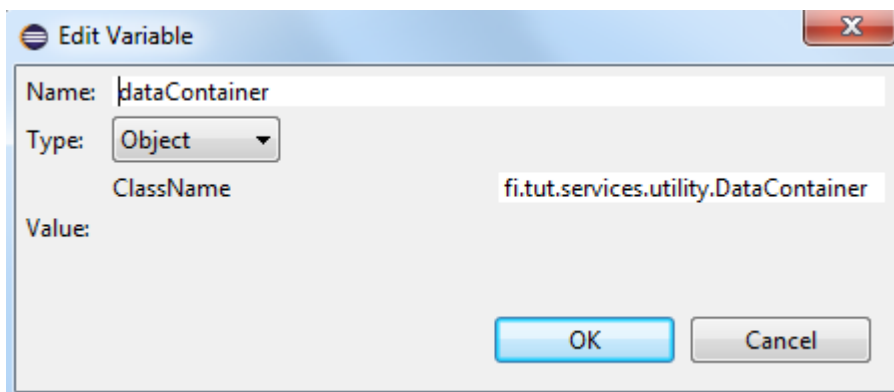


Figure 26: DataContainer set as a process variable.

Parameter Mapping	{dataContainerInput=dataContainer}
Result Mapping	{dataContainerOutput=dataContainer}

Figure 27: Parameter and result mappings of analytic service service task.

4.3 Software Project Organization

The software projects are organized into a hierarchy by using Maven project object models (POM). This allows the centralized dependency management.

The root POM defines the versions of the dependencies required by its children that are the Analytic Manager and the analytic services. Executing the Maven install on the root POM will compile the OSGi bundles of the sources and copies them to Apache ServiceMix's deploy folder as JARs (Java Archive).

5. RESULTS

This chapter presents the results of the solution that was implemented based on the following chapter. The orchestration process is followed to utilize analytic processes.

The organization of this chapter is following: first the analytic service set is introduced and then the available analytic processes are requested from the Analytic Manager. Finally the capabilities of the implemented analytic processes are demonstrated with varying scenarios. The results are evaluated in chapter 6.

5.1 System Configuration

System with following configuration was used to acquire the results:

- Microsoft Windows 7 64-bit Operating System
- Apache Tomcat 7.0 Application Server
- 8.0 GB RAM
- 4 x Intel Core i7-4600U CPU @ 2.10GHz
- JDK 1.7.0_75
- Mozilla Firefox browser

Size of Analytic Manager bundle in the file system was 183 kB.

5.2 Applied Orchestration Process

An orchestration process was defined in section (3.1). Here an implementation is proposed with following actions:

- i. Make analytic services discoverable
 - a. Analytic services are deployed to Apache ServiceMix as OSGi bundles that can be accessed via OSGi Service Registry
- ii. Utilize an orchestration engine to build and execute analytic processes
 - a. jBPM is applied to manage the analytic processes
- iii. Define how the interactions between analytic process and analytic services are managed
 - a. The solution relies on uniform interfaces defined with WSDLs and uses analytic process as mediator between the analytic services
- iv. Deploy the solution to a suitable platform and make it available for the users
 - a. Apache ServiceMix is used as an integration platform

- v. Implement a suitable client to interact with the analytic processes
 - a. A web client was implemented with Mojarra JavaServer Faces

5.3 Analytic Service Set and Service Tasks

Three Analytic Service OSGi bundles of type “SampleService” were deployed to Apache ServiceMix. They differ from each other by the quality provided for different user perspectives. These analytic services are displayed in Figure 28.

```
karaf@root> la | grep "Sample Service"
[ 203] [Active] [      ] [Started] [ 80] Sample Service :: Instance 1 :: CXF OSGi <1.0.0>
[ 204] [Active] [      ] [Started] [ 80] Sample Service :: Instance 3 :: CXF OSGi <1.0.0>
[ 205] [Active] [      ] [Started] [ 80] Sample Service :: Instance 2 :: CXF OSGi <1.0.0>
karaf@root>
```

Figure 28: Analytic Manager and Analytic Services deployed on Apache ServiceMix.

In this thesis is also used analytic services implemented in another project. They were packed into web archives (WAR) and are therefore deployed to Apache Tomcat application server. The Analytic Manager cannot discover these analytic services, which causes the need to use hard-coded URLs in their case. The analytic services deployed to Apache Tomcat are displayed in Figure 29. The functionality of the analytic services will be presented later in this chapter.















Tomcat Web Application Manager

Message:	OK				
Manager					
List Applications	HTML Manager Help	Manager Help	Server Status		
Applications					
Path	Version	Display Name	Running	Sessions	Commands
/	None specified	Welcome to Tomcat	true	0	Start Stop Reload Undeploy Expire sessions with idle ≥ 30 minutes
/ClusteringService	None specified	ClusteringService	true	0	Start Stop Reload Undeploy Expire sessions with idle ≥ 60 minutes
/CorrelationService	None specified	CorrelationService	true	0	Start Stop Reload Undeploy Expire sessions with idle ≥ 60 minutes
/FillgapService	None specified	FillgapService	true	0	Start Stop Reload Undeploy Expire sessions with idle ≥ 60 minutes
/NormalizeService	None specified	NormalizeService	true	0	Start Stop Reload Undeploy Expire sessions with idle ≥ 60 minutes
/ParetoChartService	None specified	ParetoChartService	true	0	Start Stop Reload Undeploy Expire sessions with idle ≥ 60 minutes
/RadarChartService	None specified	RadarChartService	true	0	Start Stop Reload Undeploy Expire sessions with idle ≥ 60 minutes
/SankeyService	None specified	SankeyService	true	0	Start Stop Reload Undeploy Expire sessions with idle ≥ 60 minutes

Figure 29: Analytic services deployed to Apache Tomcat.

In order to utilize the analytic services a set of service tasks were implemented in Analytic Manager. In Table 10 a short summary of the service tasks is given, including also the general ones that are not used in analytic service interactions.

Table 10: The implemented service tasks.

Category:	Name:	Icon	Description:
<i>Data Acquisition</i>	Data Acquisition		Acquires data from an external data source, e.g. Ontology Module.
	Generate Data		Generates sample data for testing purposes.
<i>Analytic services</i>	Correlation		Calculates a correlation matrix between data series.
	Clustering		Clusters the data values of the data series into performance groups.
	Fill Gaps		Uses interpolation and extrapolation functions to fill in missing data values inside data series.
	Normalization		Normalizes the data series to a given range.
	Pareto Chart		Draws a Pareto chart of the data series.
	Radar Chart		Draws a Radar chart of the data series.
	Sankey Diagram		Draws a Sankey diagram of the data series.
<i>Decision tasks</i>	User Decision		Extracts a decision parameter from the received user request.
	Process Decision		Extracts a decision parameter from the Analytic Data.
<i>Result handling</i>	Return result		Exports the processed data from the jBPM process into the Analytic Manager OSGi bundle

5.4 Implemented Analytic Processes

From the analytic service set described in the previous section a set of analytic processes were designed and packed into the Analytic Manager OSGi bundle that was deployed to Apache ServiceMix (Figure 30).

```
karaf@root> la | grep "Analytic Manager"
[ 196] [Active ] [      ] [Started] [ 80] analytic Manager <1.0.0>
karaf@root>
```

Figure 30: Analytic Manager OSGi bundle deployed to Apache Karaf.

As the Analytic Manager OSGi bundle is activated on the Apache ServiceMix it publishes the AnalyticManagerService web service with the *getAvailableProcesses* and *invokeAnalyticProcess* operations. *getAvailableProcesses* operation is used by the user to gain information of the analytic processes the Analytic Manager is managing. The web service endpoints of the solution are displayed in Figure 31.

```
karaf@root> cxf:list-endpoints
Name                State      Address
BusID
[SampleService      ] [Started ] [http://localhost:8190/cxf/AnalyticServiceSampleService_1 ]
[SampleService_instance1-cxf953775085 ]
[SampleService      ] [Started ] [http://localhost:8190/cxf/AnalyticServiceSampleService_3 ]
[SampleService_instance3-cxf1284129256 ]
[SampleService      ] [Started ] [http://localhost:8190/cxf/AnalyticServiceSampleService_2 ]
[SampleService_instance2-cxf1038284370 ]
[AnalyticManagerSOAPBinding] [Started ] [http://localhost:8190/cxf/analyticManagerService ]
[analyticModule.analyticManager-cxf1556232302]
```

Figure 31: Web service endpoints of the analytic services and the Analytic Manager deployed to Apache Karaf.

The “service address” of the Client UI was set to point to *http://localhost:8190/cxf/analyticManagerService?wsdl* and the *getAnalyticProcesses* web service was invoked by clicking the corresponding button. The web service returns an *availableProcessesResponse* message (Listing 14) that includes the basic information of the available analytic processes.

```
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<availableProcessesResponse
  xmlns="http://analytics.tut.fi/analyticManager/processesResponse">
  <analyticProcess key="fi.tut.bpmn.clustering">
    Clusters the data series regarding the user request.
  </analyticProcess>
  <analyticProcess key="fi.tut.bpmn.paretoChart">
    Returns a pareto chart drawn from the defined data
  </analyticProcess>
  <analyticProcess key="fi.tut.bpmn.sankeyChart"></analyticProcess>
  <analyticProcess key="fi.tut.bpmn.correlation"></analyticProcess>
  <analyticProcess key="fi.tut.bpmn.processDecision">
    Demonstrates process embedded decision making.
  </analyticProcess>
  <analyticProcess key="fi.tut.bpmn.userDecision">
    Demonstrates the user decision between multiple Analytic Services
    providing data visualization.
  </analyticProcess>
  <analyticProcess key="fi.tut.bpmn.radarChart1">
    Draws a radar chart from the data.
  </analyticProcess>
  <analyticProcess key="fi.tut.bpmn.radarChart2">
    Fills in missing data values, normalizes and draws a radar
    chart from the data.
  </analyticProcess>
  <analyticProcess key="fi.tut.bpmn.serviceSelection">
    This Analytic Process demonstrates the perspective-based selection
    between Analytic Service instances.
  </analyticProcess>
  <analyticProcess key="fi.tut.bpmn.stubTest1"></analyticProcess>
</availableProcessesResponse>
```

Listing 14: Result of the *getAvailableProcesses* - web service.

The concrete design of these analytic processes is presented in the next section where also their functionality is demonstrated.

5.5 Demonstration of Analytic Processes

This section uses a sample set of analytic processes to demonstrate the results provided by the analytic service orchestration.

5.5.1 Sample Data for Analytic Processes

As earlier presented in section 4.1 Analytic Manager is designed to acquire data to be used in analytic processes from the Ontology Module via a Web Service interface. However, by the time of implementation of this thesis the implementation of Ontology Module was not available. Therefore generated sets of data were used as the inputs to analytic processes.

The static set of inputDataSeries presented in Listing 15 was used to be added to the DataContainer.

```
<inputDataSeries seriesId="seriesId001">
  <metadata key="kpiName">DailyProducedProducts</metadata>
  <metadata key="consumer">ProductionLine_1</metadata>
  <metadata key="unit">pcs</metadata>
  <dataValue timestamp="1">10.0</dataValue>
  <dataValue timestamp="2">7.0</dataValue>
  <dataValue timestamp="3">5.0</dataValue>
  <dataValue timestamp="4">2.0</dataValue>
</inputDataSeries>
<inputDataSeries seriesId="seriesId002">
  <metadata key="kpiName">DailyProducedProducts</metadata>
  <metadata key="consumer">ProductionLine_2</metadata>
  <metadata key="unit">pcs</metadata>
  <dataValue timestamp="1">3.0</dataValue>
  <dataValue timestamp="2">5.0</dataValue>
  <dataValue timestamp="3">9.0</dataValue>
  <dataValue timestamp="4">6.0</dataValue>
</inputDataSeries>
<inputDataSeries seriesId="seriesId003">
  <metadata key="kpiName">AverageEnergyConsumption</metadata>
  <metadata key="consumer">Conveyor</metadata>
  <metadata key="unit">kWh</metadata>
  <dataValue timestamp="1">1.5</dataValue>
  <dataValue timestamp="2">3.0</dataValue>
  <dataValue timestamp="3">7.0</dataValue>
  <dataValue timestamp="4">9.0</dataValue>
</inputDataSeries>
<inputDataSeries seriesId="seriesId004">
  <metadata key="kpiName">AverageEnergyConsumption</metadata>
  <metadata key="consumer">Storage</metadata>
  <metadata key="unit">kWh</metadata>
  <dataValue timestamp="1">3.5</dataValue>
  <dataValue timestamp="2">4.5</dataValue>
  <dataValue timestamp="3">5.5</dataValue>
  <dataValue timestamp="4">7.0</dataValue>
</inputDataSeries>
```

Listing 15: Sample data series used to execute the Analytic Processes.

As stated before, E-Nodes were applied to present the required data to the Ontology Module. The data of the Listing 16 could be presented with corresponding E-Node structure shown in Listing 17.

```

<eNode id="ProductionLine_1">
  <data seriesId="seriesId001" type="float">
    DailyProducedProducts
  </data>
  <eNode id="Conveyor">
    <data seriesId="seriesId003" type="float">
      AverageEnergyConsumption
    </data>
  </eNode>
</eNode>
<eNode id="ProductionLine_2">
  <data seriesId="seriesId002" type="float">
    DailyProducedProducts
  </data>
</eNode>
<eNode id="ProductStorage">
  <data seriesId="seriesId004" type="float">
    AverageEnergyConsumption
  </data>
</eNode>

```

Listing 16: E-Node structure representing the sample data.

For demonstration purposes a chance was included for the user to manipulate the generated data via analytic process properties. By setting the metadata element shown in Listing 16 to the *analyticProcessRequest* message the second and third dataValue of the inputSeries002 are deleted. This allows the demonstration of the analytic service providing interpolation functions.

```

<metadata key="GenerateDataFillGaps">true</metadata>

```

Listing 17: Property to remove dataValues from a dataSeries.

5.5.2 Single Analytic Service

fi.tut.bpmn.radarChart1 analytic process contains a single analytic service. It provides the user with a service that draws a radar chart from the data defined in the user request. It is used to demonstrate the simplest case of possible analytic processes as it only contains a single analytic service. The BPMN 2.0 diagram of the *fi.tut.bpmn.radarChart1* analytic process is presented in Figure 32.

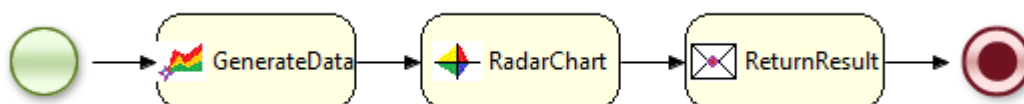


Figure 32: *fi.tut.bpmn.radarChart1* Analytic Proces BPMN diagram.

Listing 18 presents a piece of XML that contains the processParameters element with the necessary information for the Analytic Manager to execute the *fi.tut.bpmn.radarChart1* analytic process. The lines are included to the *AnalyticProcessRequest* prepared in the Client UI.

```

<processParameters>
  <!-- General process configuration -->
  <metadata key="processName">fi.tut.bpmn.radarChart1</metadata>
  <metadata key="startTime">1</metadata>
  <metadata key="endTime">4</metadata>
  <metadata key="resolutionUnit">hour</metadata>
  <metadata key="resolutionValue">1</metadata>
  <metadata key="GenerateDataFillGaps">>false</metadata>
  <!-- Radar chart specific configuration -->
  <metadata key="minRange">0</metadata>
  <metadata key="maxRange">10</metadata>
</processParameters>

```

Listing 18: processParameters element to invoke fi.tut.bpmn.radarChart1 Analytic Process

The Analytic Manager returns an AnalyticProcessResponse (Listing 19) as a response to the initial request. From the processResults element of the AnalyticProcessResponse the Client UI recognizes the metadata element containing the image information and displays it. The result is shown in Figure 33.

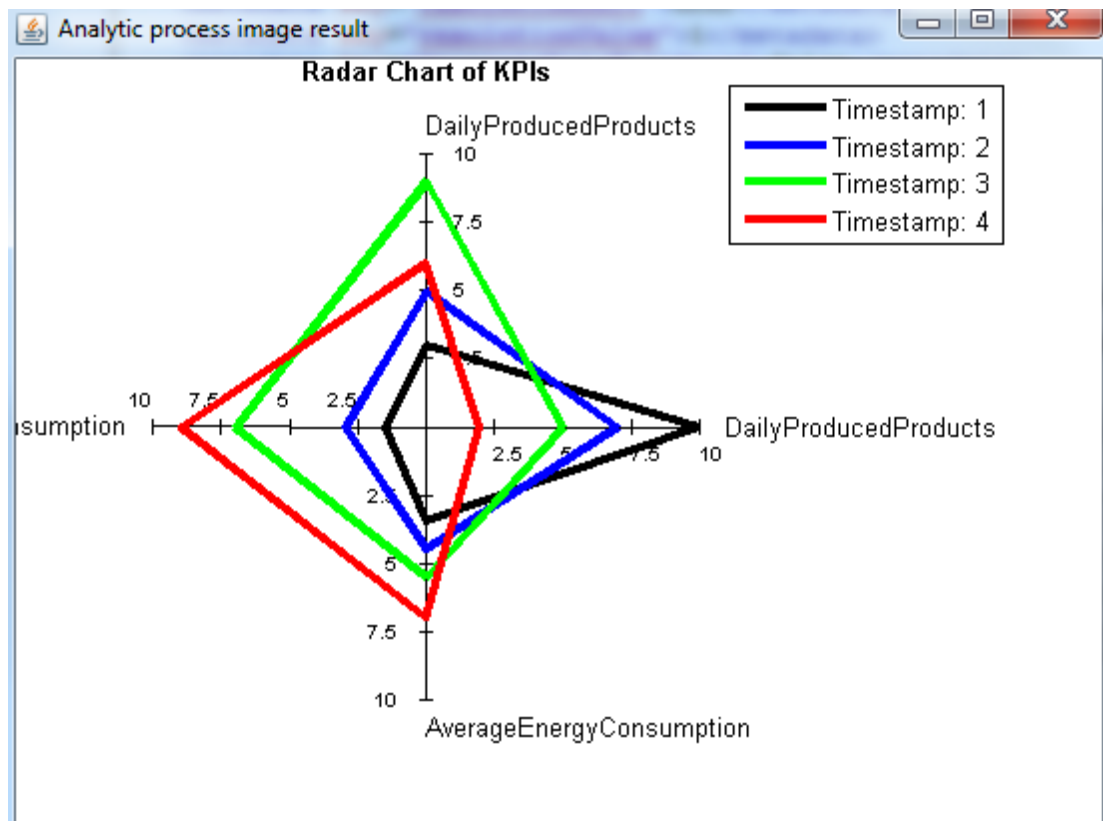


Figure 33: Result of the radarChart1 process

```

<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<analyticProcessResponse
xmlns="http://analytics.tut.fi/analyticManager/executionResponse">
  <processResults>
    <metadata key="processName">fi.tut.bpmn.radarChart1</metadata>
    <metadata key="resolutionUnit">hour</metadata>
    <metadata key="startTime">1</metadata>
    <metadata key="endTime">4</metadata>
    <metadata key="resolutionValue">1</metadata>
    <metadata key="minRange">0</metadata>
    <metadata key="maxRange">10</metadata>
    <metadata key="GenerateDataFillGaps">false</metadata>
    <metadata key="ImageOutput">
      iVBORw0KGgoAAAANSUheUgAAAjAAAAGkCAIAAACGjIjwAAAX101EQVR42u3d0ZKjOLZA
      Uf7/p7kRd2a6q2wQR9KREHit6IeOrEwbG61tYQzbDgAL2KwCAAQJAAQJAECEAEECQJAA
      QJAAECQAECQABAKA...
    </metadata>
  </processResults>
  <outputDataSeries seriesId="seriesId001">
    <metadata key="unit">kW</metadata>
    <metadata key="kpiName">DailyProducedProducts</metadata>
    <metadata key="consumer">Line_1</metadata>
    <dataValue timestamp="1">10.0</dataValue>
    <dataValue timestamp="2">7.0</dataValue>
    <dataValue timestamp="3">5.0</dataValue>
    <dataValue timestamp="4">2.0</dataValue>
  </outputDataSeries>
  <outputDataSeries seriesId="seriesId002">
    <metadata key="unit">kW</metadata>
    <metadata key="kpiName">DailyProducedProducts</metadata>
    <metadata key="consumer">Line_2</metadata>
    <dataValue timestamp="1">3.0</dataValue>
    <dataValue timestamp="2">5.0</dataValue>
    <dataValue timestamp="3">9.0</dataValue>
    <dataValue timestamp="4">6.0</dataValue>
  </outputDataSeries>
  <outputDataSeries seriesId="seriesId003">
    <metadata key="unit">kW</metadata>
    <metadata key="kpiName">AverageEnergyConsumption</metadata>
    <metadata key="consumer">Conveyor</metadata>
    <dataValue timestamp="1">1.5</dataValue>
    <dataValue timestamp="2">3.0</dataValue>
    <dataValue timestamp="3">7.0</dataValue>
    <dataValue timestamp="4">9.0</dataValue>
  </outputDataSeries>
  <outputDataSeries seriesId="seriesId004">
    <metadata key="unit">kW</metadata>
    <metadata key="kpiName">AverageEnergyConsumption</metadata>
    <metadata key="consumer">Storage</metadata>
    <dataValue timestamp="1">3.5</dataValue>
    <dataValue timestamp="2">4.5</dataValue>
    <dataValue timestamp="3">5.5</dataValue>
    <dataValue timestamp="4">7.0</dataValue>
  </outputDataSeries>
</analyticProcessResponse>

```

Listing 19: AnalyticProcessResponse containing the results of the *fi.tut.bpmn.radarChart1*.

5.5.3 Parallel Analytic Services

fi.tut.bpmn.parallelProcess analytic process demonstrates the parallel execution of analytic services (Figure 34). Two analytic services, correlation service and clustering service, use the generated data their input and the results of the services are merged to form the final result. This analytic process also combines numerical and visual results as correlation service calculates correlation coefficients between data sets and clustering service draws a diagram to illustrate the clusters of selected data sets.

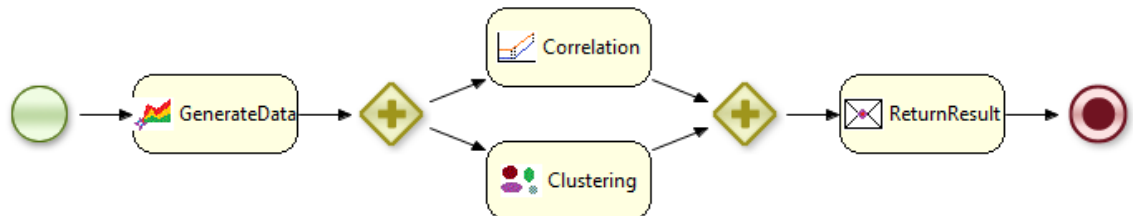


Figure 34: BPMN 2.0 diagram of *fi.tut.bpmn.parallelProcess* analytic process.

The `processParameters` presented in Listing 20 were used to invoke *fi.tut.bpmn.parallelProcess* analytic process.

```
<processParameters>
  <!-- Unbounded amount of metadata elements -->
  <metadata key='processName'>fi.tut.bpmn.parallelProcess</metadata>
  <metadata key='startTime'>1</metadata>
  <metadata key='endTime'>4</metadata>
  <metadata key='resolutionUnit'>hour</metadata>
  <metadata key='resolutionValue'>1</metadata>
  <metadata key='GenerateDataFillGaps'>>false</metadata>

  <!-- Clustering -->
  <metadata key='legendLocation'>BestOutside</metadata>
  <metadata key='pointStyle'>dot</metadata>
  <metadata key='replicates'>5</metadata>
  <metadata key='numberOfClusters'>2</metadata>
  <metadata key='xAxisData'>DailyProducedProducts</metadata>
  <metadata key='yAxisData'>AverageEnergyConsumption</metadata>

  <!-- Correlation -->
  <metadata key='maxRange'>15</metadata>
  <metadata key='minRange'>0</metadata>
</processParameters>
```

Listing 20: `ProcessParameters` of the `analyticProcessRequest`.

Following `analyticProcessResponse` is returned from the Analytic Manager after the analytic process has been invoked (Listing 21).

```

<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<analyticProcessResponse
  xmlns="http://analytics.tut.fi/analyticManager/executionResponse">
  <processResults>
    <metadata key="processName">fi.tut.bpmn.correlation</metadata>
    <metadata key="CorrCoef between Conveyor and Conveyor">
      1.0</metadata>
    <metadata key="CorrCoef between AverageEnergyConsumption and
      AverageEnergyConsumption">1.0</metadata>
    <metadata key="CorrCoef between AverageEnergyConsumption and
      Conveyor">0.9762815969903128</metadata>
    <metadata key="CorrCoef between Conveyor and
      DailyProducedProducts">-0.969304363919633</metadata>
    <metadata key="CorrCoef between DailyProducedProducts and
      AverageEnergyConsumption">-0.9947636404618068</metadata>
    <metadata key="CorrCoef between Conveyor and
      AverageEnergyConsumption">0.9762815969903128</metadata>
    <metadata key="CorrCoef between DailyProducedProducts and
      Conveyor">-0.9693043639196329</metadata>
    <metadata key="CorrCoef between DailyProducedProducts and
      DailyProducedProducts">1.0</metadata>
    <metadata key="CorrCoef between AverageEnergyConsumption and
      DailyProducedProducts">-0.9947636404618068</metadata>
    <metadata key="replicates">5</metadata>
    <metadata key="numberOfClusters">2</metadata>
    <metadata key="xAxisData">DailyProducedProducts</metadata>
    <metadata key="yAxisData">AverageEnergyConsumption</metadata>
    <metadata key="pointStyle">dot</metadata>
    <metadata key="legendLocation">BestOutside</metadata>
    <metadata key="maxRange">15</metadata>
    <metadata key="GenerateDataFillGaps">>false</metadata>
    <metadata key="resolutionValue">12</metadata>
    <metadata key="resolutionUnit">hour</metadata>
    <metadata key="ImageOutput">
      iVBORw0KGgoAAAANSUhhEUGAAAjAAAAGkCAIAAACGjIjwAAAMlk1EQVR42u3d63KjNgCA
      Ub//s+070Zl6SlkMkhAC3c6Z/ZF1vC2Rsb4ISPj8AYAGfAwBAIEAIEgCABgCABIEgA
      IEgACBIACBIAggQAaggSAIAGAIgSAAgSAIEgA...</metadata>
    </processResults>

    <outputDataSeries seriesId="seriesId001">
      <metadata key="unit">kW</metadata>
      <metadata key="kpiName">DailyProducedProducts</metadata>
      <metadata key="legendName">Line_1</metadata>
      <metadata key="consumer">Line_1</metadata>
      <dataValue timestamp="1">10.0</dataValue>
      <dataValue timestamp="2">7.0</dataValue>
      <dataValue timestamp="3">5.0</dataValue>
      <dataValue timestamp="4">2.0</dataValue>
    </outputDataSeries>
    ...
  </analyticProcessResponse>

```

Listing 21: analyticProcessResponse containing results of correlation and clustering service.

The analyticProcessResponse contains the results of the both analytic services in its processResults element. Figure 35 presents the cluster diagram extracted from the results.

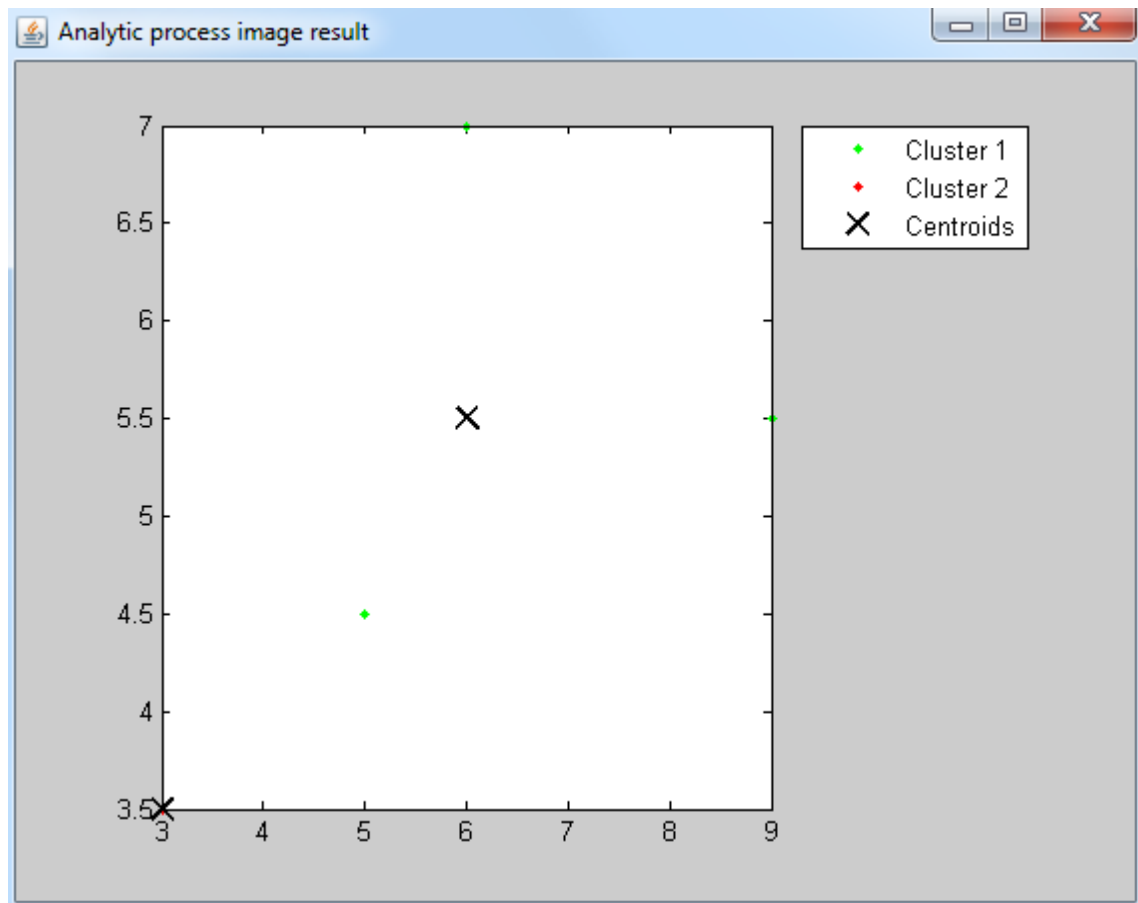


Figure 35: The image result obtained from the analyticProcessResponse.

5.5.4 Series of Analytic Services

fi.tut.bpmn.radarChart2 Analytic Process orchestrates three different analytic services into a series of operations (Figure 36). The generated analytic data is first sent to the FillGaps service that fills in the possibly missing dataValues. Then the analytic data is normalized to the range defined in the AnalyticProcessRequest. Finally the analytic data is passed to the RadarChart service that returns the radar chart decoded into a byte array.

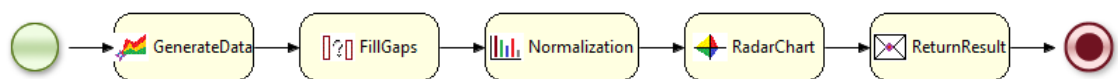


Figure 36: BPMN 2.0 presentation of *fi.tut.bpmn.radarChart2* Analytic Process

The processParameters to invoke the *fi.tut.bpmn.radarChart2* analytic process is shown in Listing 22. The instructions for specific analytic services are separated by comments. Here it is to be noted that the analytic services are configurable on behalf of the algorithmic methods and data ranges.

```

<processParameters>
  <!-- General process configuration -->
  <metadata key="processName">fi.tut.bpmn.radarChart2</metadata>
  <metadata key="startTime">1</metadata>
  <metadata key="endTime">4</metadata>
  <metadata key="resolutionUnit">hour</metadata>
  <metadata key="resolutionValue">1</metadata>
  <metadata key="GenerateDataFillGaps">true</metadata>
  <!-- Normalization specific configuration -->
  <metadata key="sequence">cross</metadata>
  <metadata key="normMin">1</metadata>
  <metadata key="normMax">5</metadata>
  <!-- Fill gaps specific configuration -->
  <metadata key="estimationMethod">2</metadata>
  <metadata key="timestampInterval">1</metadata>
  <!-- Radar chart specific configuration -->
  <metadata key="minRange">0</metadata>
  <metadata key="maxRange">5</metadata>
</processParameters>

```

Listing 22: processParameters to invoke the fi.tut.bpmn.radarChart2 Analytic Process

The AnalyticProcessResponse shown in Listing 23 is received from the Analytic Manager. The image included in the response is displayed in Figure 37.

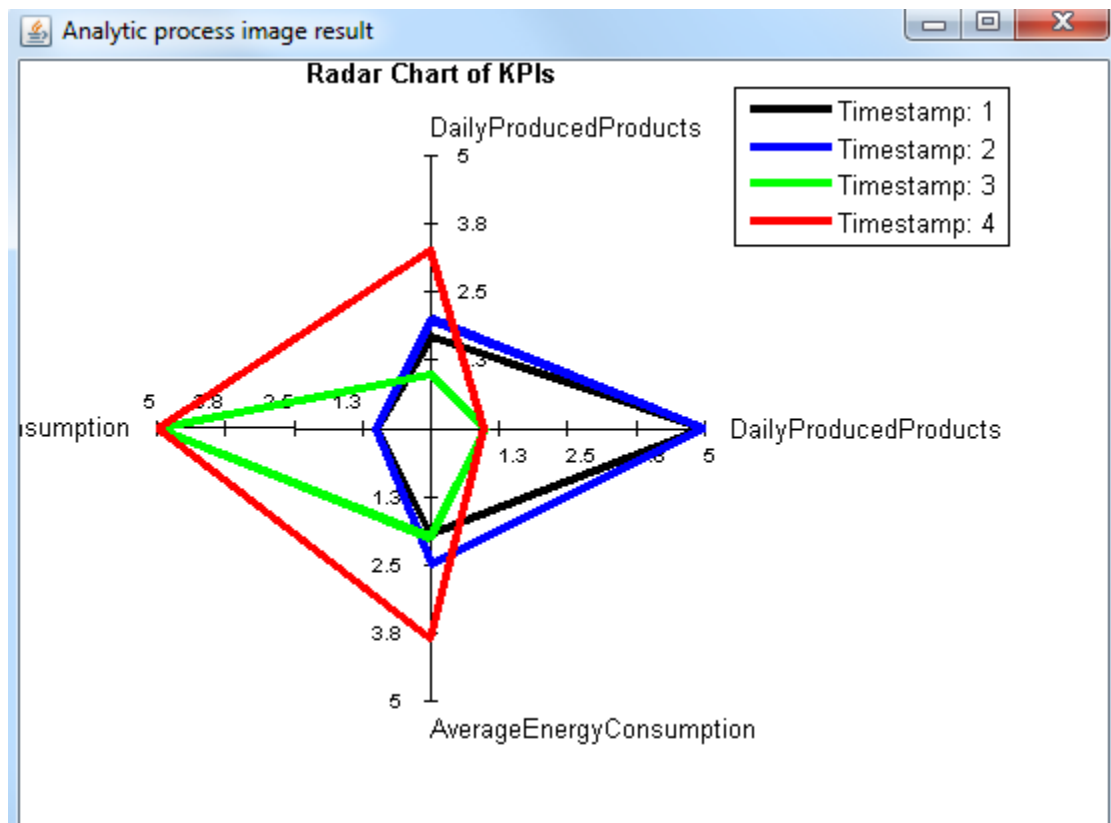


Figure 37: Result of fi.tut.bpmn.radarChart2 analytic process.

```

<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<analyticProcessResponse
xmlns="http://analytics.tut.fi/analyticManager/executionResponse">
  <processResults>
    <metadata key="processName">fi.tut.bpmn.radarChart2</metadata>
    <metadata key="minRange">0</metadata>
    <metadata key="maxRange">5</metadata>
    <metadata key="GenerateDataFillGaps">true</metadata>
    <metadata key="resolutionValue">1</metadata>
    <metadata key="startTime">1</metadata>
    <metadata key="endTime">4</metadata>
    <metadata key="resolutionUnit">hour</metadata>
    <metadata key="timestampInterval">1</metadata>
    <metadata key="normMin">1</metadata>
    <metadata key="normMax">5</metadata>
    <metadata key="estimationMethod">2</metadata>
    <metadata key="sequence">cross</metadata>
    <metadata key="ImageOutput">
      iVBORw0KGgoAAAANSUHEUgAAAjAAAAGkCAIAAACGjIjwAAAVoU1EQVR42u3d3XKjuBpA
      Ud7/pTlVc2a6ExtJn35AAtaquZhKJwZDrBlhDNsOAAvYbAIABakABakAQQIAQQJA...
    </metadata>
  </processResults>
  <outputDataSeries seriesId="seriesId001">
    <metadata key="unit">kW</metadata>
    <metadata key="kpiName">DailyProducedProducts</metadata>
    <metadata key="consumer">Line_1</metadata>
    <dataValue timestamp="1">5.0</dataValue>
    <dataValue timestamp="2">5.0</dataValue>
    <dataValue timestamp="3">1.0</dataValue>
    <dataValue timestamp="4">1.0</dataValue>
  </outputDataSeries>
  <outputDataSeries seriesId="seriesId002">
    <metadata key="unit">kW</metadata>
    <metadata key="kpiName">DailyProducedProducts</metadata>
    <metadata key="consumer">Line_2</metadata>
    <dataValue timestamp="1">1.7058823529411766</dataValue>
    <dataValue timestamp="2">2.0</dataValue>
    <dataValue timestamp="3">1.0</dataValue>
    <dataValue timestamp="4">3.2857142857142856</dataValue>
  </outputDataSeries>
  <outputDataSeries seriesId="seriesId003">
    <metadata key="unit">kW</metadata>
    <metadata key="kpiName">AverageEnergyConsumption</metadata>
    <metadata key="consumer">Conveyor</metadata>
    <dataValue timestamp="1">1.0</dataValue>
    <dataValue timestamp="2">1.0</dataValue>
    <dataValue timestamp="3">5.0</dataValue>
    <dataValue timestamp="4">5.0</dataValue>
  </outputDataSeries>
  <outputDataSeries seriesId="seriesId004">
    <metadata key="unit">kW</metadata>
    <metadata key="kpiName">AverageEnergyConsumption</metadata>
    <metadata key="consumer">Storage</metadata>
    <dataValue timestamp="1">1.9411764705882353</dataValue>
    <dataValue timestamp="2">2.5</dataValue>
    <dataValue timestamp="3">2.0</dataValue>
    <dataValue timestamp="4">3.857142857142857</dataValue>
  </outputDataSeries>
</analyticProcessResponse>

```

Listing 23: AnalyticProcessResponse to the fi.tut.bpmn.radarChart2 analytic process.

5.5.5 Demonstration of a User Decision

fi.tut.bpmn.userDecision is an analytic process that selects one of the three analytic services that each provide an visualization of the analytic data (Figure 38). The decision is based on the output of the “UserDecision” service task that extracts an execution property from the processParameters assigned by the user. Such approach makes it possible to limit the amount of required analytic processes as gateways can be used to select between specific analytic services. Also Gateways allow the designing of analytic processes that utilize the looping and branching of the execution.

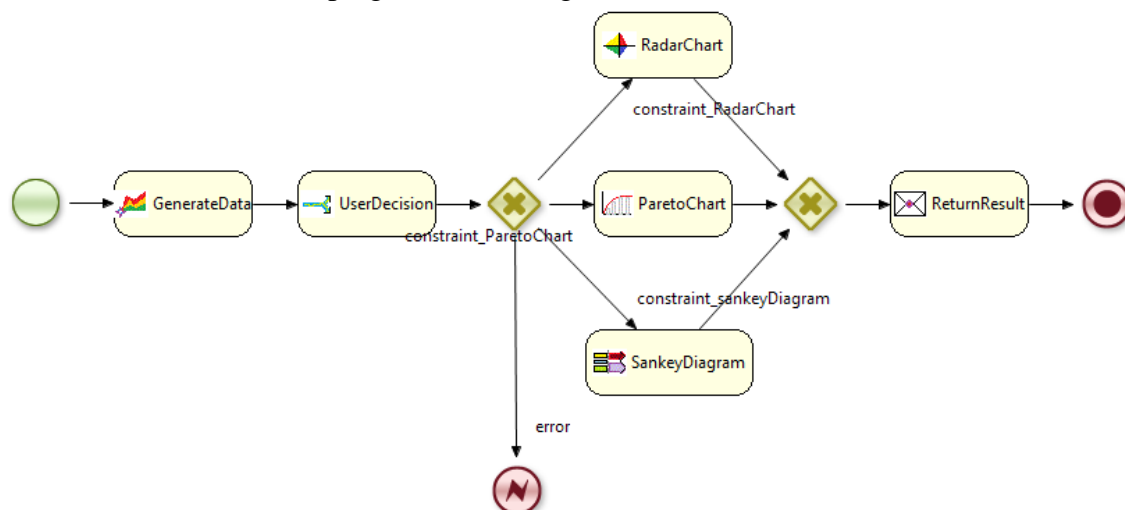


Figure 38: BPMN 2.0 presentation of *fi.tut.bpmn.userDecision* analytic process.

In Listing 24 the processParameters and its metadata elements are presented that were used to invoke the *fi.tut.bpmn.userDecision* analytic process. The metadata element with parameter *key* set to “userDecision” is used to control the selection of the analytic service providing the visualization.

```
<processParameters>
  <!-- General process configuration -->
  <metadata key="processName">
    fi.tut.bpmn.userDecisionProcess
  </metadata>
  <metadata key="startTime">1</metadata>
  <metadata key="endTime">4</metadata>
  <metadata key="resolutionUnit">hour</metadata>
  <metadata key="resolutionValue">1</metadata>
  <metadata key="GenerateDataFillGaps">false</metadata>
  <metadata key="userDecision">ParetoChart</metadata>
</processParameters>
```

Listing 24: The processParameters element used to invoke *fi.tut.bpmn.userDecision* analytic process.

The AnalyticProcessResponse presented in Listing 25 is received.

```

<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<analyticProcessResponse
xmlns="http://analytics.tut.fi/analyticManager/executionResponse">
  <processResults>
    <metadata key="resolutionUnit">hour</metadata>
    <metadata key="startTime">1</metadata>
    <metadata key="endTime">4</metadata>
    <metadata key="resolutionValue">1</metadata>
    <metadata key="userDecision">ParetoChart</metadata>
    <metadata key="processName">fi.tut.bpmn.userDecision</metadata>
    <metadata key="GenerateDataFillGaps">>false</metadata>
    <metadata key="ImageOutput">
      iVBORw0KGgoAAAANSUUhEUgAAAjAAAAGkCAIAAACGjIjwAAARvklEQVR42u3dU...
    </metadata>
  </processResults>
  <outputDataSeries seriesId="seriesId001">
    <metadata key="unit">kW</metadata>
    <metadata key="kpiName">DailyProducedProducts</metadata>
    <metadata key="consumer">Line_1</metadata>
    <dataValue timestamp="1">10.0</dataValue>
    <dataValue timestamp="2">7.0</dataValue>
    <dataValue timestamp="3">5.0</dataValue>
    <dataValue timestamp="4">2.0</dataValue>
  </outputDataSeries>
  <outputDataSeries seriesId="seriesId002">
    <metadata key="unit">kW</metadata>
    <metadata key="kpiName">DailyProducedProducts</metadata>
    <metadata key="consumer">Line_2</metadata>
    <dataValue timestamp="1">3.0</dataValue>
    <dataValue timestamp="2">5.0</dataValue>
    <dataValue timestamp="3">9.0</dataValue>
    <dataValue timestamp="4">6.0</dataValue>
  </outputDataSeries>
  <outputDataSeries seriesId="seriesId003">
    <metadata key="unit">kW</metadata>
    <metadata key="kpiName">AverageEnergyConsumption</metadata>
    <metadata key="consumer">Conveyor</metadata>
    <dataValue timestamp="1">1.5</dataValue>
    <dataValue timestamp="2">3.0</dataValue>
    <dataValue timestamp="3">7.0</dataValue>
    <dataValue timestamp="4">9.0</dataValue>
  </outputDataSeries>
  <outputDataSeries seriesId="seriesId004">
    <metadata key="unit">kW</metadata>
    <metadata key="kpiName">AverageEnergyConsumption</metadata>
    <metadata key="consumer">Storage</metadata>
    <dataValue timestamp="1">3.5</dataValue>
    <dataValue timestamp="2">4.5</dataValue>
    <dataValue timestamp="3">5.5</dataValue>
    <dataValue timestamp="4">7.0</dataValue>
  </outputDataSeries>
</analyticProcessResponse>

```

Listing 25: AnalyticProcessResponse to fi.tut.bpmn.userDecision analytic process.

The Pareto chart image included in the response is presented in Figure 39.

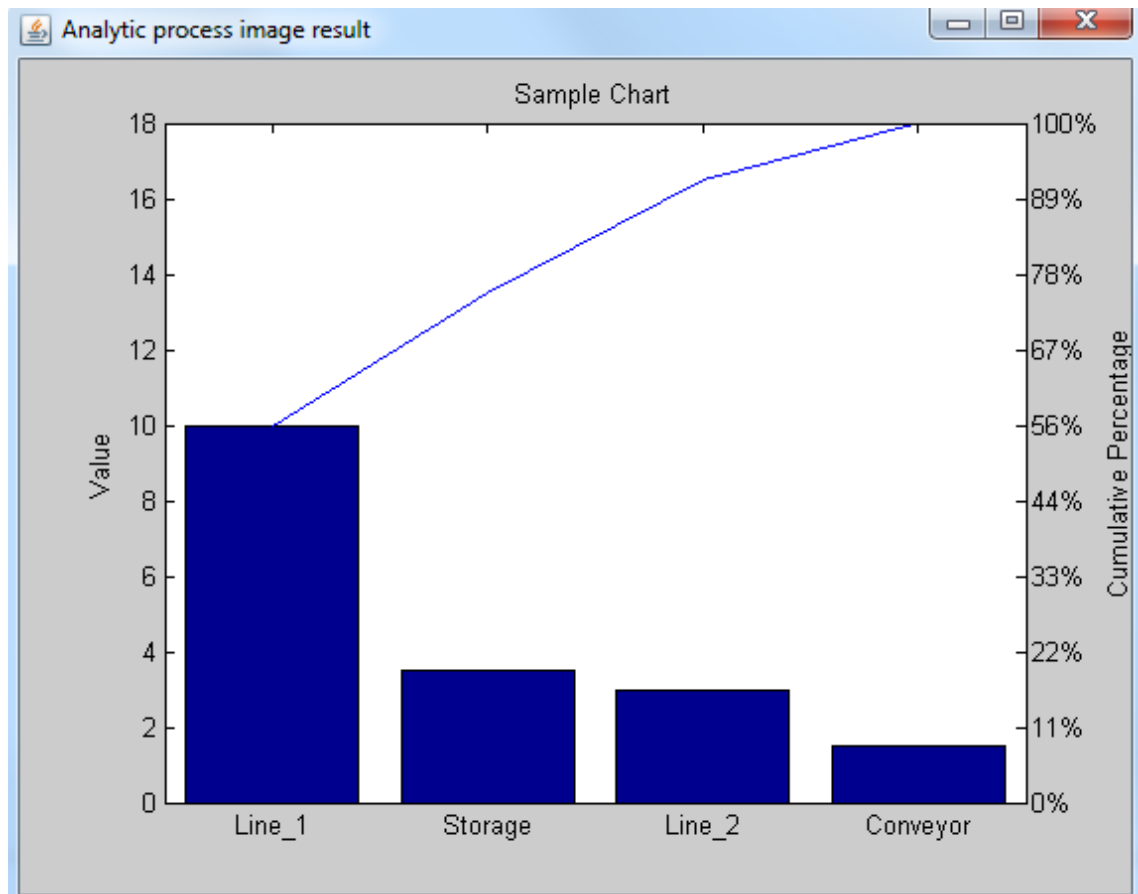


Figure 39: Pareto chart image returned from the `fi.tut.bpmn.userDecisionProcess` analytic process.

5.5.6 Demonstration of a Process Decision

Another type of provided decision types is process decision. In `fi.tut.bpmn.processDecision` analytic process the `ProcessDecision` service task is used to determine if `FillGaps` analytic service needs to be included in the execution or not (Figure 40). Within `ProcessDecision` service task an algorithm checks if the analytic data has missing data values in its data series. Therefore the use of `FillGaps` analytic service is controlled on base of the analytic data and embedded algorithms, hence allowing the optimization of the analytic process execution.

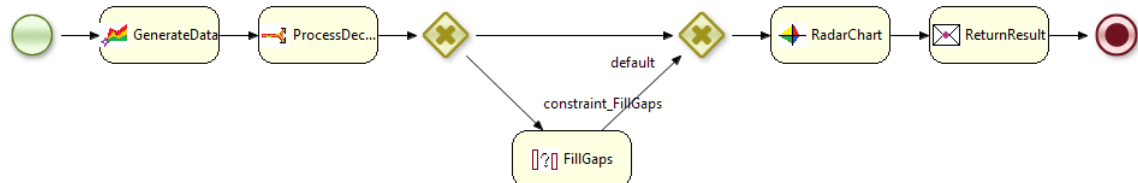


Figure 40: BPMN 2.0 presentation of `fi.tut.bpmn.processDecision` analytic process.

For the diverging Gateway a constraint is given that depends on `decisionValue` process variable. If the `decisionValue` equals with “FillGaps” the `FillGaps` service task is executed. The Constraint editor shown in Figure 41 presents the configuration of “constraint_FillGaps”. If the `decisionValue` does not match with “FillGaps”, the default execution continues to the converging Gateway and therefore bypasses the `FillGaps` service task.

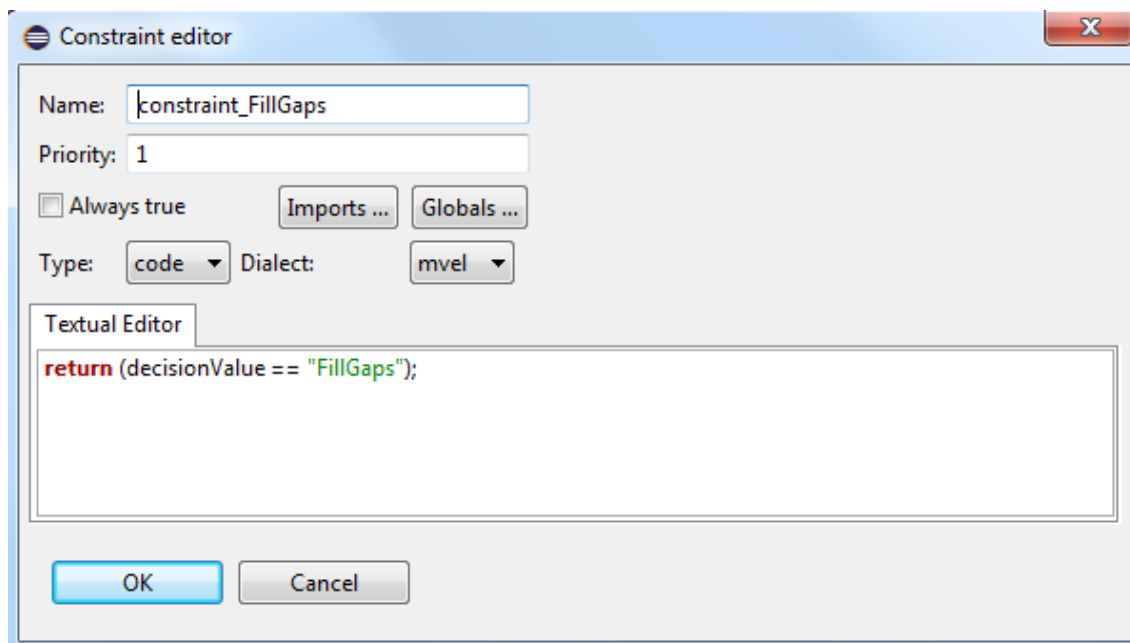


Figure 41: Configuration of the Gateway constraint.

In order to demonstrate the decision support based on the analytic data the *fi.tut.bpmn.processDecision* was invoked with two different data value sets: a full data value set and a data value set with missing values.

fi.tut.bpmn.processDecision with full data set

In the first case the value of the **GenerateDataFillGaps** property is changed to **false** in the processParameters (Listing 26), which adds missing data values to the analytic data.

```

<processParameters>
  <!-- General process configuration -->
  <metadata
key="processName">fi.tut.bpmn.processDecision</metadata>
  <metadata key="startTime">1</metadata>
  <metadata key="endTime">4</metadata>
  <metadata key="resolutionUnit">hour</metadata>
  <metadata key="resolutionValue">1</metadata>
  <metadata key="GenerateDataFillGaps">false</metadata>
  <!-- Fill gaps specific configuration -->
  <metadata key="estimationMethod">2</metadata>
  <metadata key="timestampInterval">1</metadata>
  <!-- Radar chart specific configuration -->
  <metadata key="minRange">0</metadata>
  <metadata key="maxRange">10</metadata>
</processParameters>

```

Listing 26: processParameters to invoke the analytic process with expected data values.

As the analytic process execution is invoked the dataSeries are delivered to the ProcessDecision service task with all expected data values. Therefore the *decisionValue* process variable remains null and the *FillGaps* service task is not executed.

The Listing 27 displays the AnalyticServiceResponse received from the Analytic Manager.

```

<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<analyticProcessResponse
xmlns="http://analytics.tut.fi/analyticManager/executionResponse">
  <processResults>
    <metadata key="resolutionUnit">hour</metadata>
    <metadata key="startTime">1</metadata>
    <metadata key="timestampInterval">1</metadata>
    <metadata key="estimationMethod">2</metadata>
    <metadata key="maxRange">5</metadata>
    <metadata key="processName">fi.tut.bpmn.processDecision</metadata>
    <metadata key="minRange">0</metadata>
    <metadata key="GenerateDataFillGaps">>false</metadata>
    <metadata key="resolutionValue">1</metadata>
    <metadata key="endTime">4</metadata>
    <metadata key="ImageOutput">
      iVBORw0KGgoAAAANSUhEUgAAAjAAAAGkCAIAAACGjIjwAAAYiU1EQVR42u3di3Kj2BVA
      Uf7/p0klk3G3Jbjc94u1K6nq8tgSwhLLByE4TkmSJuiwCiRJQJIKC...
    </metadata>
  </processResults>
  ...
  <outputDataSeries seriesId="seriesId002">
    <metadata key="unit">kW</metadata>
    <metadata key="kpiName">DailyProducedProducts</metadata>
    <metadata key="consumer">Line_2</metadata>
    <dataValue timestamp="1">3.0</dataValue>
    <dataValue timestamp="2">5.0</dataValue>
    <dataValue timestamp="3">9.0</dataValue>
    <dataValue timestamp="4">6.0</dataValue>
  </outputDataSeries>
  ...
</analyticProcessResponse>

```

Listing 27: Contents of the received AnalyticProcessResponse.

The ImageOutput bytes are transformed into an image object that is displayed in Figure 42.

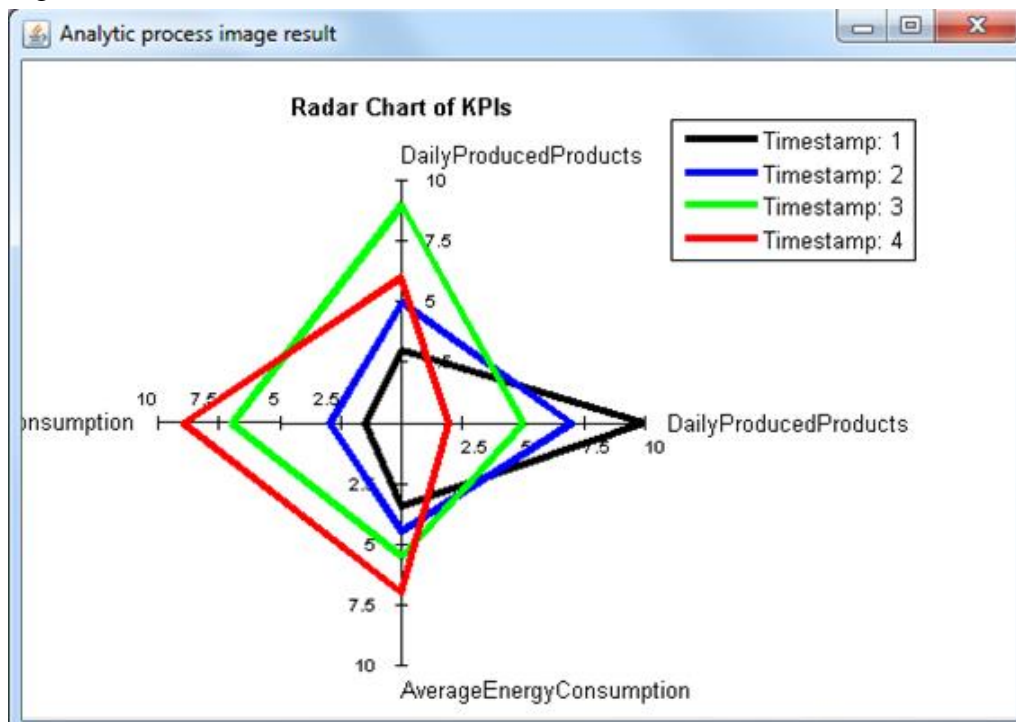


Figure 42: Radar chart figure included in the AnalyticProcessResponse.

fi.tut.bpmn.processDecision with missing data values

Next the same analytic process was invoked with the `GenerateDataFillGaps` property set to `true`. The processParameters are shown in Listing 28.

```
<processParameters>
  <!-- General process configuration -->
  <metadata
key="processName">fi.tut.bpmn.processDecision</metadata>
  <metadata key="startTime">1</metadata>
  <metadata key="endTime">4</metadata>
  <metadata key="resolutionUnit">hour</metadata>
  <metadata key="resolutionValue">1</metadata>
  <metadata key="GenerateDataFillGaps">true</metadata>
  <!-- Fill gaps specific configuration -->
  <metadata key="estimationMethod">2</metadata>
  <metadata key="timestampInterval">1</metadata>
  <!-- Radar chart specific configuration -->
  <metadata key="minRange">0</metadata>
  <metadata key="maxRange">10</metadata>
</processParameters>
```

Listing 28: processParameters to invoke the analytic process with missing data values.

The Analytic Manager returns the `AnalyticServiceResponse` presented in Listing 29. Now as there are missing dataValues in the `inputDataSeries002` the `ProcessDecision` service task recognizes this and sets the `decisionValue` process variable to “FillGaps” value. The `FillGaps` service task is executed and the missing dataValues replaced with estimated values.

```

<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<analyticProcessResponse
xmlns="http://analytics.tut.fi/analyticManager/executionResponse">
  <processResults>
    <metadata key="processName">fi.tut.bpmn.processDecision</metadata>
    <metadata key="startTime">1</metadata>
    <metadata key="endTime">4</metadata>
    <metadata key="timestampInterval">1</metadata>
    <metadata key="resolutionUnit">hour</metadata>
    <metadata key="resolutionValue">1</metadata>
    <metadata key="estimationMethod">2</metadata>
    <metadata key="minRange">0</metadata>
    <metadata key="maxRange">5</metadata>
    <metadata key="GenerateDataFillGaps">true</metadata>
    <metadata key="ImageOutput">
      iVBORw0KGgoAAAANSUhEUgAAAjAAAAGkCAIAAACGjIjwAAAY/01EQVR42u3di26jSBqA
      Ud7/pVlpZrY7sYui7hc4n3a1Vk8HiGNz8mMMxy1J0gIdHgJJEPAkSQKSJAlIkiQBSZ...
    </metadata>
  </processResults>
  ...
  <outputDataSeries seriesId="seriesId002">
    <metadata key="unit">kW</metadata>
    <metadata key="kpiName">DailyProducedProducts</metadata>
    <metadata key="consumer">Line_2</metadata>
    <dataValue timestamp="1">3.0</dataValue>
    <dataValue timestamp="2">4.0</dataValue>
    <dataValue timestamp="3">5.0</dataValue>
    <dataValue timestamp="4">6.0</dataValue>
  </outputDataSeries>
  ...
</analyticProcessResponse>

```

Listing 29: The received AnalyticProcessResponse.

The Client UI extracts the image data from the AnalyticProcessResponse and displays it (Figure 43).

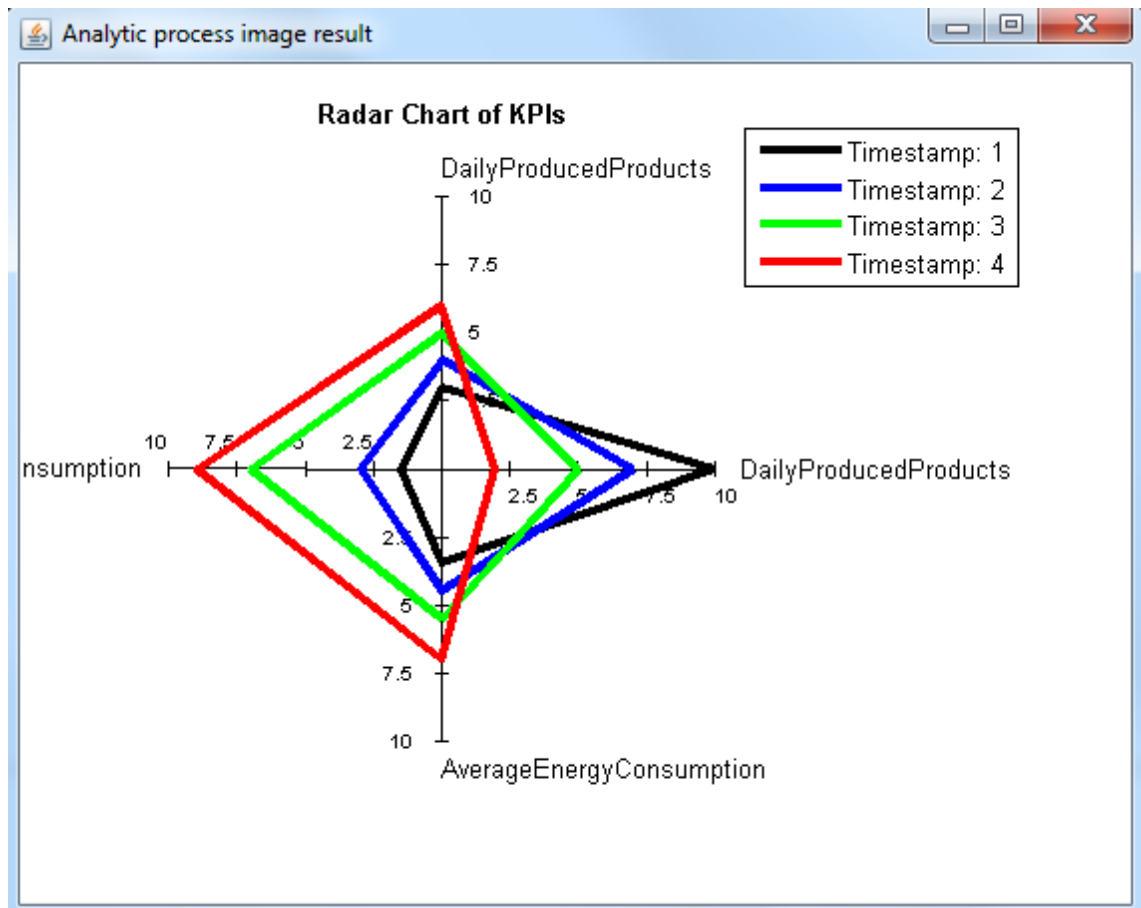


Figure 43: The image result extracted from the `AnalyticProcessResponse`.

5.5.7 Demonstration of Service Selection

fi.tut.bpmn.serviceSelection (Figure 44) is used to demonstrate the execution of an analytic process that has multiple instances of required analytic services available. From the available analytic service instances the one offering the best quality for the user with a specific perspective is selected.

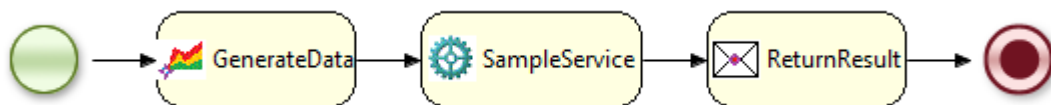


Figure 44: BPMN 2.0 presentation of *fi.tut.bpmn.serviceSelection* analytic process.

Three instances of the `SampleService` analytic service are deployed on the Apache ServiceMix at the same time with different web service endpoints. They have no business functionality as they only copy the content of the `AnalyticServiceRequest` to the `AnalyticServiceResponse`. Each instance of `SampleService` has been assigned with different quality information as described in the following (Listing 30, Listing 31 and Listing 32).

```

<wsdl:documentation>
  Quality information of sampleService_1:
  [perspective1
  [price:2.0,accuracy:4.0,time:3.0,availability:6.0];
  perspective2[price:3.0,accuracy:4.0,time:3.5,availability:0.0]]
</wsdl:documentation>

```

Listing 30: Quality information of the SampleService Instance_1.

```

<wsdl:documentation>
  Quality information of sampleService_2:
  [perspective1[price:4.0,accuracy:5.0]]
</wsdl:documentation>

```

Listing 31: Quality information of the SampleService Instance_2.

```

<wsdl:documentation>
  Quality information of sampleService_3:
  [perspective1[price:7.0,accuracy:5.0,availability:6.5];
  perspective2[price:6.0,accuracy:4.0,availability:7.0]]
</wsdl:documentation>

```

Listing 32: Quality information of the SampleService Instance_3.

The *fi.tut.bpmn.serviceSelection* is invoked with the processParameters presented in Listing 33.

```

<processParameters>
  <!-- General process configuration -->
  <metadata
  key="processName">fi.tut.bpmn.serviceSelection</metadata>
  <metadata key="startTime">1</metadata>
  <metadata key="endTime">4</metadata>
  <metadata key="resolutionUnit">hour</metadata>
  <metadata key="resolutionValue">1</metadata>
  <metadata key="GenerateDataFillGaps">true</metadata>
  <!-- Perspective related configuration -->
  <metadata key="perspective">perspective1</metadata>
  <metadata key="qualityAttributes">
    perspective1[price:-0.1,accuracy:4,time:5]</metadata>
</processParameters>

```

Listing 33: processParameters containing the perspective specific quality parameter weights.

Analytic Manager calculates the quality values for each available SampleService instance:

$$\text{Instance}_1 = -0.1 \cdot 2 + 4 \cdot 4 + 5 \cdot 3 = 30.8$$

$$\text{Instance}_2 = -0.1 \cdot 4 + 4 \cdot 5 + 5 \cdot 0 = 19.6$$

$$\text{Instance}_3 = -0.1 \cdot 7 + 4 \cdot 5 + 5 \cdot 0 = 19.3$$

Therefore the instance1 has the highest quality value for perspective1 regarding the SampleService and Analytic Manager chooses it for the execution. After finishing the analytic process an analyticProcessResponse element is returned with following value

within the processResults element meaning that the Instance_1 of SampleService was executed (Listing 34).

```
<metadata key="sampleService">Instance_1</metadata>
```

Listing 34: Received information of the selected analytic service.

Next the same analytic process was invoked with the processParameters presented in Listing 35.

```
<processParameters>
  <!-- General process configuration -->
  <metadata
    key="processName">fi.tut.bpmn.serviceSelection</metadata>
  <metadata key="startTime">1</metadata>
  <metadata key="endTime">4</metadata>
  <metadata key="resolutionUnit">hour</metadata>
  <metadata key="resolutionValue">1</metadata>
  <metadata key="GenerateDataFillGaps">true</metadata>
  <!-- Perspective related configuration -->
  <metadata key="perspective">perspective2</metadata>
  <metadata key="qualityAttributes">perspective2 [price:-
    0.6,accuracy:2.0,availability:3.5] </metadata>
</processParameters>
```

Listing 35: processParameters element to invoke the analytic process for "perspective2".

In this case Analytic Manager acquires the quality information for “perspective2” to calculate the quality values for the available SampleService instances:

$$\text{Instance}_1 = -0.6*3.0+2.0*4.0+3.5*0.0 = 6.2$$

$$\text{Instance}_2 = -0.6*0.0+2.0*0.0+3.5*0.0 = 0.0$$

$$\text{Instance}_3 = -0.6*6.0+2.0*4.0+3.5*7.0 = 28.9$$

Instance_3 is determined to possess the highest quality value for “perspective2” and it is selected. In the AnalyticProcessResponse the metadata element presented in Listing 36 is contained in processResults element:

```
<metadata key="sampleService">Instance_3</metadata>
```

Listing 36: Received information of the selected analytic service.

6. CONCLUSION

This chapter presents the conclusion of this thesis work by reviewing the initial problem and the proposed solution.

This thesis presented an approach of how analytic services can be orchestrated into analytic processes in holistic energy management systems. A set of analytic processes that presented possible tools for monitoring, visualization and decision support to be used in EMS were demonstrated. The orchestration was designed on top of SOA architecture and applied the BPM methodology to manage the analytic process execution. The implementation relies on uniform interfaces between the system components and allows their flexible reuse based on the user requirements.

The gained results present the flexibility and functional scalability of the BPM-based management of analytic services. Rapid deployment of new analytic processes is fast due to the reusability of analytic services. From the business point of view the service-based approach can enhance the development of EMS as the markets can provide an effective set of analytic services for the industrial applications.

The presented approach applies the common Internet technologies that enhance the accessibility of the provide Analytic Manager component and the availability of the analytic services. The Analytic Manager was implemented as an OSGi bundle that was deployed to Apache ServiceMix ESB. ESB is gaining popularity as an integration platform for enterprise IT applications and can greatly improve the cooperation of enterprise software. Implementation of EMS analytics in an ESB interoperable manner makes it a component that can be integrated into existing enterprise systems.

One of the main goals of this thesis was to include the concept of holism to the EMS analytics, which was taken into account when designing the solution's architecture and handling of user perspectives. The increased measurement data caused by the IoT, system integration and the support of user ontologies can be seen as enablers of holism. On the base of the literature review done in chapter 2 the used means can be seen as essential requirements for holism.

6.1 Future Work

This section proposes some possible ways to continue the presented research.

The implementation did not include the Ontology Module, as only static data was used. In future the Analytic Manager should be applied with the Ontology Module in order to enable the use of real factory data. The Ontology Module would also allow the

integration of the quality perspective into the data acquisition. With the access to real data the real feasibility of the proposed design could be tested.

The analytic services that provided real business value were only accessible with static IP addresses without support for dynamic service discovery. The analytic services should either be implemented as OSGi bundles or then some means of dynamic discovery should be used to make them visible on the ESB.

The Client UI was only a simplified user interface that allowed the demonstration of the Analytic Manager capabilities. For real use the GUI should provide means of storing and comparing the results. Also the user perspectives should be included into the GUI design to support different user types.

REFERENCES

- [1]: Europe 2020 targets. [WWW] [Accessed on 8.5.2015] Available at: http://ec.europa.eu/europe2020/europe-2020-in-a-nutshell/targets/index_en.htm
- [2]: Arinez, J.; Biller, S. “Integration requirements for manufacturing-based Energy Management Systems”, *Innovative Smart Grid Technologies (ISGT)*, 2010
- [3]: Singh, D.; Tripathi, G.; Jara, A.J. “A survey of Internet-of-Things: Future vision, architecture, challenges and services”, *Internet of Things (WF-IoT), 2014 IEEE World Forum*, 2014
- [4]: Kalagiakos, P.; Karampelas, P. “Cloud computing learning”. *5th International Conference on Information and Communication Technologies (AICT)*, 2011
- [5]: Jongwoo Choi ; Youn Kwae Jeong ; Il Woo Lee. “A Building Energy Management System Based on Facility Sensor Networks”, *Information Science and Applications (ICISA)*, 2014
- [6]: Veleva, S.; Davcev, D. “Data mining as an enabling technology for Home Energy Management System” *Innovative Smart Grid Technologies (ISGT), 2012 IEEE PES 2012*, Page(s): 1 – 8
- [7]: May, G.; Taisch, M.; Kelly, D. “Enhanced Energy Management In Manufacturing Through Systems Integration”, *Industrial Electronics Society, IECON 2013 - 39th Annual Conference*, 2013, Page(s): 7525 – 7530
- [8]: Florea, A. ; Postelnicu, C. ; Martinez Lastra, J.L. ; Presser, M. ; Plambech, T. ; Larrañaga, M. ; Colino, A. ; Márquez Contreras, J.A. ; Bayona Pons, V.M. “Decision support tool for retrofitting a district towards district as a service”, *IEEE International Workshop on Intelligent Energy Systems (IWIES)*, 2013, Page(s): 70 – 75
- [9]: Khosrow Ebrahimi, Gerard F. Jones, Amy S. Fleischer. “A review of data center cooling technology, operating conditions and the corresponding low-grade waste heat recovery opportunities”, 2013
- [10]: Florea, Anna; Postelnicu, Corina; Zhang, Bin; Martinez Lastra, Jose Luis. “Ecosystem oriented energy management: An implementation”, *IEEE International Conference on Systems, Man, and Cybernetics (SMC)*, 2012
- [11]: Shamszaman, Z.U.; Sanghong Lee; Ilyoung Chong. “WoO based user centric Energy Management System in the internet of things”, *International Conference on Information Networking (ICOIN)*, 2014.

- [12]: Mora, D.; Taisch, M.; Colombo, A.W.; Mendes, J.M. “Service-oriented architecture approach for industrial “System of Systems”: State-of-the-Art for Energy Management”, *10th IEEE International Conference on Industrial Informatics (INDIN)*, 2012, Page(s): 1246 - 1251
- [13]: ISA-95 Website [WWW] [Accessed on 8.5.2015] Available at: <http://www.isa-95.com/>
- [14]: Batch Control Website. [WWW] [Accessed on 8.5.2015] Available at: <http://www.batchcontrol.com/s95/s95.shtml>
- [15]: Vasyutynskyy, V.; Hengstler, C.; McCarthy, J.; Brennan, K.G.; Nadoveza, D.; Dennert, A. “Layered architecture for production and logistics cockpits”, *IEEE 17th Conference on Emerging Technologies & Factory Automation (ETFA)*, 2012
- [16]: Florea, A.; Montemayor, J.A.G.I.; Postelnicu, C.; Lastra, J.L.M. “A cross-layer approach to energy management in manufacturing”, *10th IEEE International Conference on Industrial Informatics (INDIN)*, 2012
- [17]: Brenda M. Michelson. “Event-driven Architecture Overview”, 2006, [WWW] [Accessed on 8.5.2015] Available at: [www.omg.org/soa/Uploaded Docs/EDA/bda2-2-06cc.pdf](http://www.omg.org/soa/UploadedDocs/EDA/bda2-2-06cc.pdf)
- [18]: Santoro, R.; Braccini, A.; Vecchi, C.; Fantini, C. “Odysseus — Open Dynamic System for holistic energy management — Rome XI district case study: The “Municipality of Rome” XI odysseus pilot cases case study: Technical features, requirements and constraints for improvement and optimization of the energy system Rome municipality neighborhoods”, *International ICE, Engineering, Technology and Innovation (ICE)*, 2014.
- [19]: Kurpatov, R.; Schmidt, M.; Schulke, A. “Enabling complex building energy visualization by integrative event-driven data provisioning”, *International Conference and Workshops on Networked Systems (NetSys)*, 2015
- [20]: Dandan Liu; Qijun Chen. “Prediction of building lighting energy consumption based on support vector regression”, *9th Asian Control Conference (ASCC)*, 2013
- [21]: Veleva, S.; Davcev, D. “Data mining as an enabling technology for Home Energy Management System”, *Innovative Smart Grid Technologies (ISGT)*, 2012.
- [22]: Mohamed, A.; Salehi, V.; Tan Ma; Mohammed, O. “Real-Time Energy Management Algorithm for Plug-In Hybrid Electric Vehicle Charging Parks Involving Sustainable Energy”. *IEEE Transactions on Sustainable Energy*, 2014
- [23]: Elsied, M., Oukaour, A. ; Gualous, H. ; Hassan, R. ; Amin, A. “An Advanced Energy Management of Microgrid System Based on Genetic Algorithm”, *IEEE 23rd International Symposium on Industrial Electronics (ISIE)*, 2014

- [24]: Martirano, L.; Manganelli, M.; Parise, L.; Sbordone, D.A. "Design of a fuzzy-based control system for energy saving and users comfort", *14th International Conference on Environment and Electrical Engineering (EEEIC)*, 2014
- [25]: Casali, A.; Ernst, C. "Discovering Correlated Parameters in Semiconductor Manufacturing Processes: A Data Mining Approach", *Transactions on Semiconductor Manufacturing*, 2012
- [26]: Xindong Wu; Xingquan Zhu; Gong-Qing Wu; Wei Ding. "Data mining with big data", *IEEE Transactions on Knowledge and Data Engineering*, 2014
- [27]: Yang, Gao; Tumwesigye, E.; Cahill, B.; Menzel, K. "Using data mining in optimisation of building energy consumption and thermal comfort management", *2nd International Conference on Software Engineering and Data Mining (SEDM)*, 2010
- [28]: SQL Data Mining Website. [WWW] [Accessed on 8.5.2015] Available at: <http://www.sqldatamining.com/index.php/data-warehousing/steps-of-the-knowledge-discovery-in-databases-process>
- [29]: Alexandra Instituttet Website. [WWW] [Accessed on 8.5.2015] Available at: <http://alexandra.dk/dk/cases/urb-grade>
- [30]: Reference Model for Service Oriented Architecture 1.0, OASIS, [WWW] [Accessed on 8.5.2015] Available at: <http://docs.oasis-open.org/soa-rm/v1.0/soa-rm.html>
- [31]: Valipour, M.H. ; Amirzafari, B. ; Maleki, K.N. ; Daneshpour, N. "A brief survey of software architecture concepts and service oriented architecture" *2nd IEEE International Conference on Computer Science and Information Technology*, 2009 , Page(s): 34 – 38
- [32]: Nagorny, K.; Harrison, R.; Colombo, A.W.; Kreutz, G. "A formal engineering approach for control and monitoring systems in a service-oriented environment", *11th IEEE International Conference on Industrial Informatics (INDIN)*, 2013
- [33]: M. T. Schmidt, etc., "The Enterprise Service Bus: Making service-oriented architecture real, " *IBM Systems Journal, VOL 44, NO 4*, 2005, pp. 781-797
- [34]: IBM Website. [WWW] [Accessed on 8.5.2015] Available at: <http://www.ibm.com/developerworks/library/ar-esbpat1/>
- [35]: Christensen, J.; Strasser, T.; Valentini, A.; Vyatkin, V.; Zoitl, A. "IEC 61499-1: Function Blocks - Part 1 Architecture", *International Electrotechnical Commission, International Standard, First Edition*, 2005
- [36]: Hästbacka, D.; Kuikka, S. "Facilitating services and engineering process management in distributed engineering of control applications", *10th IEEE Industrial Informatics (INDIN)*, 2012

- [37]: Reference Architecture for Service-Oriented Architecture Version 1.0, OASIS public review draft. 2008, [WWW] [Accessed on 8.5.2015] Available at: <http://docs.oasis-open.org/soa-rm/soa-ra/v1.0/soa-ra-pr-01.pdf>
- [38]: Web Services Architecture W3C Working Group Note 11 February 2004 [WWW] [Accessed on 8.5.2015] Available at: <http://www.w3.org/TR/ws-arch/#whatis>
- [39]: Kiran Kumar Reddy D; Thirumaran, M; Karthiek Maralla; Raj Kumar G. “A Greedy Approach with Criteria Factors for QoS based Web Service Discovery”, *COMPUTE '09*, ISBN: 978-1-60558-476-8, 2009
- [40]: WSDL 1.1 structure. [WWW] [Accessed on 8.5.2015] Available at: http://inchoo.net/wp-content/uploads/2012/04/wsdl_objects.jpg
- [41]: Chinnici, Roberto; Moreau, Jean-Jacques; Ryman, Arthur; Weerawarana, Sanjiva. “Web Services Description Language (WSDL) Version 2.0 Part 1: Core Language”, *W3C Recommendation 26 June 2007*, [WWW] [Accessed on 8.5.2015] Available at: <http://www.w3.org/TR/wsd120/>
- [42]: SOAP structure. [WWW] [Accessed on 8.5.2015] Available at: <http://docs.oracle.com/cd/E19798-01/821-1796/aeqex/index.html>
- [43]: Davis, A.; Du Zhang. “A comparative study of DCOM and SOAP”. *Fourth International Symposium on Multimedia Software Engineering*, 2002
- [44]: JSR 222: Java™ Architecture for XML Binding (JAXB) 2.0. [WWW] [Accessed on 8.5.2015] Available at: <https://jcp.org/en/jsr/detail?id=222>
- [45]: JSR 224: Java™ API for XML-Based Web Services (JAX-WS) 2.0. [WWW] [Accessed on 8.5.2015] Available at: <https://jcp.org/en/jsr/detail?id=224>
- [46]: OSGi Alliance homepage. [WWW] [Accessed on 8.5.2015] Available at: <http://www.osgi.org/Main/HomePage>
- [47]: (WS09-01) Management von OSGi-basierten Anwendungen. [WWW] [Accessed on 8.5.2015] Available at: https://wwwvs.cs.hs-rm.de/vs-wiki/index.php/%28WS09-01%29_Management_von_OSGi-basierten_Anwendungen
- [48]: OSGi architecture. [WWW] [Accessed on 8.5.2015] Available at: <http://www.osgi.org/Technology/WhatIsOSGi>
- [49]: Figure of OSGi bundle lifecycle. [WWW] [Accessed on 8.5.2015] Available at: <http://devangelist.blogspot.fi/2011/04/osgi-bundle-lifecycles.html>
- [50]: Apache Karaf Website. [WWW] [Accessed on 8.5.2015] Available at: <http://karaf.apache.org/>
- [51]: Jiang Ji-chen; Gao Ming. “Enterprise Service Bus and an Open Source Implementation”, *International Conference on Management Science and Engineering, ICMSE '06*, 2006.

- [52]: Figure of Apache ServiceMix's components. [WWW] [Accessed on 8.5.2015] Available at: <http://www.cnblogs.com/ajian005/archive/2012/05/24/2753731.html>
- [53]: Van der Aalst, W; Hofstede, A; Weske, Mathias. "Business Process Management: A Survey", 2003.
- [54]: Business Process Model and Notation (BPMN) Version 2.0. [WWW] [Accessed on 8.5.2015] Available at: <http://www.omg.org/spec/BPMN/2.0/> 2. November 2014.
- [55]: jBPM 5.4 user guide. [WWW] [Accessed on 8.5.2015] Available at: <http://docs.jboss.org/jbpm/v5.1/userguide/>
- [56]: Drools Website. [WWW] [Accessed on 8.5.2015] Available at: <http://www.drools.org/>
- [57]: Maven Website. [WWW] [Accessed on 8.5.2015] Available at: <http://maven.apache.org/>
- [58]: Spring Framework Website. [WWW] [Accessed on 8.5.2015] Available at: <http://projects.spring.io/spring-framework/>. 3. November, 2014.
- [59]: Mukherjee, A.; Tari, Z.; Bertok, P. "A Spring Based Framework for Verification of Service Composition", *IEEE International Conference on Services Computing (SCC)*, 2011.
- [60]: Mojarra JavaServer Faces Website. [WWW] [Accessed on 8.5.2015] Available at: <https://javaserverfaces.java.net/>
- [61]: Gosling, J.; Joy, B.; Steele, G; Bracha, G.; Buckley, A. "The Java Language Specification: Java SE 7 Edition", 2013, [WWW] [Accessed on 8.5.2015] Available at: <http://docs.oracle.com/javase/specs/jls/se7/html/index.html>
- [62]: Apache Tomcat web site. [WWW] [Accessed on 8.5.2015] Available at: <https://tomcat.apache.org/download-70.cgi>
- [63]: Apache CXF Website. [WWW] [Accessed on 8.5.2015] Available at: <http://cxf.apache.org/>
- [64]: Spring Web Service Website. [WWW] [Accessed on 8.5.2015] Available at: <http://projects.spring.io/spring-ws/>
- [65]: Mateos, C.; Crasso, M.; Zunino, A.; Coscia, J.L.O. "Revising WSDL Documents: Why and How, Part 2", *Internet Computing*. 2013
- [66]: Mule ESB Website. [WWW] [Accessed on 12.5.2015] Available at: <http://www.mulesoft.org/what-mule-esb>
- [67]: Apache ServiceMix Website. [WWW] [Accessed on 12.5.2015] Available at: <http://servicemix.apache.org/>

[68]: WSO2 ESB Website. [WWW] [Accessed on 12.5.2015] Available at:
<http://wso2.com/products/enterprise-service-bus/>

[69]: Petals ESB Website. [WWW] [Accessed on 12.5.2015] Available at:
<http://petals.ow2.org/>

APPENDIX A: SYSTEM CONFIGURATION

This section gives general information about the system configuration.

OSGi bundles deployed on ServiceMix 4.5.3

Following OSGi bundles were deployed in order to satisfy the dependencies of Analytic Manager OSGi bundle:

- antlr-runtime-3.5.jar
- drools-compiler-5.5.0.Final.jar
- drools-core-5.5.0.Final.jar
- jbpm-bpmn2-5.4.0.Final.jar
- jbpm-flow-5.4.0.Final.jar
- jbpm-flow-builder-5.4.0.Final.jar
- knowledge-api-5.5.0.Final.jar
- knowledge-internal-api-5.5.0.Final.jar
- mvel2-2.1.0.Final.jar
- protobuf-java-2.5.0.jar
- servicemix-common-2013.01.jar

Development environment

Kepler version of Eclipse IDE was used in the development with following plugins:

- m2e 1.4.0
- Spring Tool Suite 3.4.0.RELEASE
- Apache CXF version 2.7.5
- jBPM 5.4.0 Eclipse plugin

APPENDIX B: XML INTERFACES

This section presents the XSDs of the messages and the WSDLs used in the solution.

List of contents:

- analyticProcessRequest.xsd
- analyticProcessResponse.xsd
- availableProcessesRequest.xsd
- availableProcessesResponse.xsd
- analyticServiceRequest.xsd
- analyticServiceResponse.xsd
- dataRequest.xsd
- dataResponse.xsd
- analyticManager.wsdl
- sampleService1.wsdl
- sampleService2.wsdl
- sampleService3.wsdl

analyticProcessRequest.xsd:

```
<?xml version="1.0" encoding="UTF-8"?>
<schema xmlns="http://www.w3.org/2001/XMLSchema"
  xmlns:xs="http://www.w3.org/2001/XMLSchema"
  xmlns:tns="http://analytics.tut.fi/analyticManager/executionRequest"
  targetNamespace="http://analytics.tut.fi/analyticManager/executionRequest"
  elementFormDefault="qualified">

  <!-- Definition of dataRequest element, the base element of the schema -->
  <xs:element name="analyticProcessRequest">
    <xs:complexType>
      <xs:sequence>
        <xs:element name="processParameters">
          <xs:complexType>
            <xs:sequence>
              <xs:element ref="tns:metadata" maxOccurs="unbounded"/>
            </xs:sequence>
          </xs:complexType>
        </xs:element>
        <xs:element ref="tns:eNode" maxOccurs="unbounded"/>
      </xs:sequence>
    </xs:complexType>
  </xs:element>

  <!-- Definition of metadata element -->
  <xs:element name="metadata">
```

```

<xs:complexType>
  <xs:simpleContent>
    <xs:extension base="xs:string">
      <xs:attribute name="key" type="xs:string"/></xs:attribute>
    </xs:extension>
  </xs:simpleContent>
</xs:complexType>
</xs:element>

<!-- Definition of eNode element -->
<xs:element name="eNode">
  <xs:complexType>
    <xs:sequence>
      <xs:element ref="tns:dataId" minOccurs="0" maxOccurs="unbounded"/>
      <xs:element ref="tns:eNode" minOccurs="0" maxOccurs="unbounded"/>
    </xs:sequence>
    <xs:attribute name="id" type="xs:string"/></xs:attribute>
  </xs:complexType>
</xs:element>

<!-- Definition of dataId element -->
<xs:element name="dataId">
  <xs:complexType>
    <xs:simpleContent>
      <xs:extension base="xs:string">
        <xs:attribute name="type" type="xs:string"/></xs:attribute>
        <xs:attribute name="seriesId" type="xs:string"/></xs:attribute>
      </xs:extension>
    </xs:simpleContent>
  </xs:complexType>
</xs:element>

</schema>

```

analyticProcessResponse.xsd:

```

<?xml version="1.0" encoding="UTF-8"?>
<schema xmlns="http://www.w3.org/2001/XMLSchema"
  xmlns:xs="http://www.w3.org/2001/XMLSchema"
  xmlns:tns="http://analytics.tut.fi/analyticManager/executionResponse"
  targetNamespace="http://analytics.tut.fi/analyticManager/executionResponse"
  elementFormDefault="qualified">

  <!-- Definition of analyticProcessResponse element,
  the base element of the schema -->
  <xs:element name="analyticProcessResponse">
    <xs:annotation>
      <xs:documentation>
        analyticProcessResponse contains the data and parameters received from
        analytic process execution
      </xs:documentation>
    </xs:annotation>
    <xs:complexType>
      <xs:sequence>
        <xs:element ref="tns:processResults" minOccurs="1" maxOccurs="1"/>
        <xs:element ref="tns:outputDataSeries" maxOccurs="unbounded"/>
      </xs:sequence>
    </xs:complexType>
  </xs:element>

```



```

<!-- Definition of processResults element -->
<xs:element name="processResults">
  <xs:annotation>
    <xs:documentation>
      processResults element contains all the parameters to be received from
      the analytic process
    </xs:documentation>
  </xs:annotation>
  <xs:complexType>
    <xs:sequence>
      <!-- metadata elements to describe the processResults -->
      <xs:element ref="tns:metadata" maxOccurs="unbounded"/>
    </xs:sequence>
  </xs:complexType>
</xs:element>

<!-- Definition of outputDataSeries element -->
<xs:element name="outputDataSeries">
  <xs:annotation>
    <xs:documentation>
      outputDataSeries contains metadata and concrete data of a certain data
      entity
    </xs:documentation>
  </xs:annotation>
  <xs:complexType>
    <xs:sequence>
      <!-- metadata elements to describe the inputDataSeries -->
      <xs:element ref="tns:metadata" maxOccurs="unbounded"/>
      <!-- dataValues of the inputDataSeries -->
      <xs:element ref="tns:dataValue" maxOccurs="unbounded"/>
    </xs:sequence>
    <xs:attribute name="seriesId" type="xs:string"/>
  </xs:complexType>
</xs:element>

<!-- Definition of metadata element -->
<xs:element name="metadata">
  <xs:annotation>
    <xs:documentation>
      metadata is used to store key-value pairs that describe process results
      and data series
    </xs:documentation>
  </xs:annotation>
  <xs:complexType>
    <xs:simpleContent>
      <xs:extension base="xs:string">
        <xs:attribute name="key" type="xs:string" use="required"/>
      </xs:extension>
    </xs:simpleContent>
  </xs:complexType>
</xs:element>

<!-- Definition of dataValue element -->
<xs:element name="dataValue">
  <xs:annotation>
    <xs:documentation>
      A set of data value elements is used to store data values with their cor
      responding timestamps
    </xs:documentation>
  </xs:annotation>
  </xs:element>

```

```

    </xs:documentation>
  </xs:annotation>
</xs:complexType>
  <xs:simpleContent>
    <xs:extension base="xs:string">
      <xs:attribute name="timestamp" type="xs:long" use="required"/>
    </xs:extension>
  </xs:simpleContent>
</xs:complexType>
</xs:element>

</schema>

```

availableProcessesRequest.xsd:

```

<?xml version="1.0" encoding="UTF-8"?>
<schema xmlns="http://www.w3.org/2001/XMLSchema"
  xmlns:xs="http://www.w3.org/2001/XMLSchema"
  xmlns:tns="http://analytics.tut.fi/analyticManager/processesRequest"
  targetNamespace="http://analytics.tut.fi/analyticManager/processesRequest"
  elementFormDefault="qualified">

  <!-- Definition of availableProcessesRequest element, the base element of
  the schema -->
  <xs:element name="availableProcessesRequest">
    <xs:annotation>
      <xs:documentation>
        availableProcessesRequest contains the parameters defined by the
        client
      </xs:documentation>
    </xs:annotation>
    <xs:complexType>
      <xs:sequence>
        <xs:element ref="tns:requestParameters" minOccurs="1" maxOccurs="1"/>
      </xs:sequence>
    </xs:complexType>
  </xs:element>

  <!-- Definition of requestParameters element -->
  <xs:element name="requestParameters">
    <xs:annotation>
      <xs:documentation>
        requestParameters element contains all the parameters to be received from
        the client
      </xs:documentation>
    </xs:annotation>
    <xs:complexType>
      <xs:sequence>
        <!-- metadata elements to describe the requestParameters -->
        <xs:element ref="tns:metadata" maxOccurs="unbounded"/>
      </xs:sequence>
    </xs:complexType>
  </xs:element>

  <!-- Definition of metadata element -->
  <xs:element name="metadata">
    <xs:annotation>
      <xs:documentation>
        metadata is used to store key-value pairs that describe the

```

```

    request</xs:documentation>
  </xs:annotation>
</xs:complexType>
  <xs:simpleContent>
    <xs:extension base="xs:string">
      <xs:attribute name="key" type="xs:string" use="required"/>
    </xs:extension>
  </xs:simpleContent>
</xs:complexType>
</xs:element>

</schema>

```

availableProcessesResponse.xsd:

```

<?xml version="1.0" encoding="UTF-8"?>
<schema xmlns="http://www.w3.org/2001/XMLSchema"
  xmlns:xs="http://www.w3.org/2001/XMLSchema"
  xmlns:tns="http://analytics.tut.fi/analyticManager/processesResponse"
  targetNamespace="http://analytics.tut.fi/analyticManager/processesResponse"
  elementFormDefault="qualified">

  <!-- Definition of availableProcessesRequest element, the base element of
  the schema -->
  <xs:element name="availableProcessesResponse">
    <xs:annotation>
      <xs:documentation>
        availableProcessesResponse contains descriptions of the available
        processes
      </xs:documentation>
    </xs:annotation>
    <xs:complexType>
      <xs:sequence>
        <xs:element ref="tns:analyticProcess" maxOccurs="unbounded"/>
      </xs:sequence>
    </xs:complexType>
  </xs:element>

  <!-- Definition of analyticProcess element -->
  <xs:element name="analyticProcess">
    <xs:annotation>
      <xs:documentation>
        analyticProcess is used to store key-value pairs that describe the
        analytic process
      </xs:documentation>
    </xs:annotation>
    <xs:complexType>
      <xs:simpleContent>
        <xs:extension base="xs:string">
          <xs:attribute name="key" type="xs:string" use="required"/>
        </xs:extension>
      </xs:simpleContent>
    </xs:complexType>
  </xs:element>

</schema>

```

analyticServiceRequest.xsd:

```

<?xml version="1.0" encoding="UTF-8"?>

```

```

<schema xmlns="http://www.w3.org/2001/XMLSchema"
  xmlns:xs="http://www.w3.org/2001/XMLSchema"
  xmlns:tns="http://analytics.tut.fi/analyticServices/request/schema"
  targetNamespace="http://analytics.tut.fi/analyticServices/request/schema"
  elementFormDefault="qualified">

  <!-- Definition of analyticServiceRequest element, the base element of the
  schema -->
  <xs:element name="analyticServiceRequest">
    <xs:annotation>
      <xs:documentation>
        analyticServiceRequest contains the data and parameters required for
        analytic service execution
      </xs:documentation>
    </xs:annotation>
    <xs:complexType>
      <xs:sequence>
        <xs:element ref="tns:serviceParameters" minOccurs="1" maxOccurs="1"/>
        <xs:element ref="tns:inputDataSeries" maxOccurs="unbounded"/>
      </xs:sequence>
    </xs:complexType>
  </xs:element>

  <!-- Definition of serviceParameters element -->
  <xs:element name="serviceParameters">
    <xs:annotation>
      <xs:documentation>
        serviceParameters element contains all the parameters to be passed to the
        analytic service
      </xs:documentation>
    </xs:annotation>
    <xs:complexType>
      <xs:sequence>
        <!-- metadata elements to describe the serviceParameters -->
        <xs:element ref="tns:metadata" maxOccurs="unbounded"/>
      </xs:sequence>
    </xs:complexType>
  </xs:element>

  <!-- Definition of inputDataSeries element -->
  <xs:element name="inputDataSeries">
    <xs:annotation>
      <xs:documentation>
        inputDataSeries contains metadata and concrete data of a certain data
        entity
      </xs:documentation>
    </xs:annotation>
    <xs:complexType>
      <xs:sequence>
        <!-- metadata elements to describe the inputDataSeries -->
        <xs:element ref="tns:metadata" maxOccurs="unbounded"/>
        <!-- dataValues of the inputDataSeries -->
        <xs:element ref="tns:dataValue" maxOccurs="unbounded"/>
      </xs:sequence>
      <xs:attribute name="seriesId" type="xs:string"/></xs:attribute>
    </xs:complexType>
  </xs:element>

  <!-- Definition of metadata element -->

```

```

<xs:element name="metadata">
  <xs:annotation>
    <xs:documentation>
      metadata is used to store key-value pairs that describe service
      parameters and data series
    </xs:documentation>
  </xs:annotation>
  <xs:complexType>
    <xs:simpleContent>
      <xs:extension base="xs:string">
        <xs:attribute name="key" type="xs:string" use="required"/>
      </xs:extension>
    </xs:simpleContent>
  </xs:complexType>
</xs:element>

<!-- Definition of dataValue element -->
<xs:element name="dataValue">
  <xs:annotation>
    <xs:documentation>
      A set of data value elements is used to store data values with their
      corresponding timestamps
    </xs:documentation>
  </xs:annotation>
  <xs:complexType>
    <xs:simpleContent>
      <xs:extension base="xs:string">
        <xs:attribute name="timestamp" type="xs:Long" use="required"/>
      </xs:extension>
    </xs:simpleContent>
  </xs:complexType>
</xs:element>

</schema>

```

analyticServiceResponse.xsd:

```

<?xml version="1.0" encoding="UTF-8"?>
<schema xmlns="http://www.w3.org/2001/XMLSchema"
  xmlns:xs="http://www.w3.org/2001/XMLSchema"
  xmlns:tns="http://analytics.tut.fi/analyticServices/response/schema"
  targetNamespace="http://analytics.tut.fi/analyticServices/response/schema"
  elementFormDefault="qualified">

  <!-- Definition of analyticServiceResponse element, the base element of the
  schema -->
  <xs:element name="analyticServiceResponse">
    <xs:annotation>
      <xs:documentation>
        analyticServiceResponse contains the data and parameters received from
        analytic service execution
      </xs:documentation>
    </xs:annotation>
    <xs:complexType>
      <xs:sequence>
        <xs:element ref="tns:serviceResults" minOccurs="1" maxOccurs="1"/>
        <xs:element ref="tns:outputDataSeries" maxOccurs="unbounded"/>
      </xs:sequence>
    </xs:complexType>

```

```

</xs:element>

<!-- Definition of serviceResults element -->
<xs:element name="serviceResults">
  <xs:annotation>
    <xs:documentation>
      serviceResults element contains all the parameters to be received from
      the analytic service
    </xs:documentation>
  </xs:annotation>
  <xs:complexType>
    <xs:sequence>
      <!-- metadata elements to describe the serviceResults -->
      <xs:element ref="tns:metadata" maxOccurs="unbounded"/>
    </xs:sequence>
  </xs:complexType>
</xs:element>

<!-- Definition of outputDataSeries element -->
<xs:element name="outputDataSeries">
  <xs:annotation>
    <xs:documentation>
      outputDataSeries contains metadata and concrete data of a certain data
      entity
    </xs:documentation>
  </xs:annotation>
  <xs:complexType>
    <xs:sequence>
      <!-- metadata elements to describe the inputDataSeries -->
      <xs:element ref="tns:metadata" maxOccurs="unbounded"/>
      <!-- dataValues of the inputDataSeries -->
      <xs:element ref="tns:dataValue" maxOccurs="unbounded"/>
    </xs:sequence>
    <xs:attribute name="seriesId" type="xs:string"/>
  </xs:complexType>
</xs:element>

<!-- Definition of metadata element -->
<xs:element name="metadata">
  <xs:annotation>
    <xs:documentation>
      metadata is used to store key-value pairs that describe service results
      and data series
    </xs:documentation>
  </xs:annotation>
  <xs:complexType>
    <xs:simpleContent>
      <xs:extension base="xs:string">
        <xs:attribute name="key" type="xs:string" use="required"/>
      </xs:extension>
    </xs:simpleContent>
  </xs:complexType>
</xs:element>

<!-- Definition of dataValue element -->
<xs:element name="dataValue">
  <xs:annotation>
    <xs:documentation>
      A set of data value elements is used to store data values with their

```

```

    corresponding timestamps
  </xs:documentation>
</xs:annotation>
<xs:complexType>
  <xs:simpleContent>
    <xs:extension base="xs:string">
      <xs:attribute name="timestamp" type="xs:long" use="required"/>
    </xs:extension>
  </xs:simpleContent>
</xs:complexType>
</xs:element>

</schema>

```

dataRequest.xsd:

```

<?xml version="1.0" encoding="UTF-8"?>
<schema xmlns="http://www.w3.org/2001/XMLSchema"
  xmlns:xs="http://www.w3.org/2001/XMLSchema"
  xmlns:tns="http://analytics.tut.fi/dataAcquisition/request/schema"
  targetNamespace="http://analytics.tut.fi/dataAcquisition/request/schema"
  elementFormDefault="qualified">

  <!-- Definition of dataRequest element, the base element of the schema -->
  <xs:element name="dataRequest">
    <xs:complexType>
      <xs:sequence>
        <xs:element name="requestParameters">
          <xs:complexType>
            <xs:sequence>
              <xs:element ref="tns:metadata" maxOccurs="unbounded"/>
            </xs:sequence>
          </xs:complexType>
        </xs:element>
        <xs:element ref="tns:eNode" maxOccurs="unbounded"/>
      </xs:sequence>
    </xs:complexType>
  </xs:element>

  <!-- Definition of metadata element -->
  <xs:element name="metadata">
    <xs:complexType>
      <xs:simpleContent>
        <xs:extension base="xs:string">
          <xs:attribute name="key" type="xs:string"/>
        </xs:extension>
      </xs:simpleContent>
    </xs:complexType>
  </xs:element>

  <!-- Definition of eNode element -->
  <xs:element name="eNode">
    <xs:complexType>
      <xs:sequence>
        <xs:element ref="tns:dataId" minOccurs="0" maxOccurs="unbounded"/>
        <xs:element ref="tns:eNode" minOccurs="0" maxOccurs="unbounded"/>
      </xs:sequence>
      <xs:attribute name="id" type="xs:string"/>
    </xs:complexType>
  </xs:element>

```

```

</xs:element>

<!-- Definition of dataId element -->
<xs:element name="dataId">
  <xs:complexType>
    <xs:simpleContent>
      <xs:extension base="xs:string">
        <xs:attribute name="type" type="xs:string"></xs:attribute>
        <xs:attribute name="seriesId" type="xs:string"></xs:attribute>
      </xs:extension>
    </xs:simpleContent>
  </xs:complexType>
</xs:element>

</schema>

```

dataResponse.xsd:

```

<?xml version="1.0" encoding="UTF-8"?>
<schema xmlns="http://www.w3.org/2001/XMLSchema"
  xmlns:xs="http://www.w3.org/2001/XMLSchema"
  xmlns:tns="http://analyticsservices.tut.fi/dataAcquisition/response/schema"
  targetNamespace=
    "http://analyticsservices.tut.fi/dataAcquisition/response/schema"
  elementFormDefault="qualified">

  <!-- Definition of analyticServiceRequest element, the base element of the
  schema -->
  <xs:element name="dataResponse">
    <xs:complexType>
      <xs:sequence>
        <xs:element ref="tns:responseParameters" minOccurs="1" maxOccurs="1"/>
        <xs:element ref="tns:responseDataSeries" maxOccurs="unbounded"/>
      </xs:sequence>
    </xs:complexType>
  </xs:element>

  <!-- Definition of responseParameters element -->
  <xs:element name="responseParameters">
    <xs:complexType>
      <xs:sequence>
        <!-- metadata elements to describe the serviceParameters -->
        <xs:element ref="tns:metadata" maxOccurs="unbounded"/>
      </xs:sequence>
    </xs:complexType>
  </xs:element>

  <!-- Definition of responseDataSeries element -->
  <xs:element name="responseDataSeries">
    <xs:complexType>
      <xs:sequence>
        <!-- metadata elements to describe the inputDataSeries -->
        <xs:element ref="tns:metadata" maxOccurs="unbounded"/>
        <!-- dataValues of the inputDataSeries -->
        <xs:element ref="tns:dataValue" maxOccurs="unbounded"/>
      </xs:sequence>
      <xs:attribute name="seriesId" type="xs:string"></xs:attribute>
    </xs:complexType>
  </xs:element>

```



```

<!-- Definition of metadata element -->
<xs:element name="metadata">
  <xs:complexType>
    <xs:simpleContent>
      <xs:extension base="xs:string">
        <xs:attribute name="key" type="xs:string" use="required"/>
      </xs:extension>
    </xs:simpleContent>
  </xs:complexType>
</xs:element>

<!-- Definition of dataValue element -->
<xs:element name="dataValue">
  <xs:complexType>
    <xs:simpleContent>
      <xs:extension base="xs:string">
        <xs:attribute name="timestamp" type="xs:Long" use="required"/>
      </xs:extension>
    </xs:simpleContent>
  </xs:complexType>
</xs:element>

</schema>

```

analyticManager.wsdl:

```

<wsdl:definitions
  xmlns:soap="http://schemas.xmlsoap.org/wsdl/soap/"
  xmlns:tns="http://analytics.tut.fi/analyticManager"
  xmlns:execreqns="http://analytics.tut.fi/analyticManager/executionRequest"
  xmlns:execrespns="http://analytics.tut.fi/analyticManager/executionResponse"
  xmlns:processesreqns=
    "http://analytics.tut.fi/analyticManager/processesRequest"
  xmlns:processesrespns=
    "http://analytics.tut.fi/analyticManager/processesResponse"
  xmlns:wsdl="http://schemas.xmlsoap.org/wsdl/"
  xmlns:xsd="http://www.w3.org/2001/XMLSchema"
  targetNamespace="http://analytics.tut.fi/analyticManager">

  <wsdl:types>
    <schema xmlns=http://www.w3.org/2001/XMLSchema
      xmlns:xs="http://www.w3.org/2001/XMLSchema"
      elementFormDefault="qualified"
      targetNamespace="http://analytics.tut.fi/analyticManager">
      <xs:import
        namespace="http://analytics.tut.fi/analyticManager/executionRequest"
        schemaLocation="messageSchemas/analyticProcessRequest.xsd">
      </xs:import>
      <xs:import
        namespace="http://analytics.tut.fi/analyticManager/executionResponse"
        schemaLocation="messageSchemas/analyticProcessResponse.xsd">
      </xs:import>
      <xs:import
        namespace="http://analytics.tut.fi/analyticManager/processesRequest"
        schemaLocation="messageSchemas/availableProcessesRequest.xsd">
      </xs:import>
      <xs:import
        namespace="http://analytics.tut.fi/analyticManager/processesResponse"

```

```

        schemaLocation="messageSchemas/availableProcessesResponse.xsd">
    </xs:import>
</schema>
</wsdl:types>

<wsdl:message name="analyticProcessRequest">
    <wsdl:part element="execreqns:analyticProcessRequest"
        name="analyticProcessRequest">
    </wsdl:part>
</wsdl:message>
<wsdl:message name="analyticProcessResponse">
    <wsdl:part element="execrespns:analyticProcessResponse"
        name="analyticProcessResponse">
    </wsdl:part>
</wsdl:message>
<wsdl:message name="availableProcessesRequest">
    <wsdl:part element="processesreqns:availableProcessesRequest"
        name="availableProcessesRequest">
    </wsdl:part>
</wsdl:message>
<wsdl:message name="availableProcessesResponse">
    <wsdl:part element="processesrespns:availableProcessesResponse"
        name="availableProcessesResponse">
    </wsdl:part>
</wsdl:message>

<wsdl:portType name="AnalyticManager">
    <wsdl:operation name="invokeAnalyticProcess">
        <wsdl:input message="tns:analyticProcessRequest"
            name="analyticProcessRequest"/>
        <wsdl:output message="tns:analyticProcessResponse"
            name="analyticProcessResponse"/>
    </wsdl:operation>
    <wsdl:operation name="getAvailableProcesses">
        <wsdl:input message="tns:availableProcessesRequest"
            name="availableProcessesRequest"/>
        <wsdl:output message="tns:availableProcessesResponse"
            name="availableProcessesResponse"/>
    </wsdl:operation>
</wsdl:portType>

<wsdl:binding name="AnalyticManagerSOAPBinding" type="tns:AnalyticManager">
    <soap:binding style="document"
        transport="http://schemas.xmlsoap.org/soap/http"/>
    <wsdl:operation name="invokeAnalyticProcess">
        <wsdl:input name="analyticProcessRequest">
            <soap:body use="literal"/>
        </wsdl:input>
        <wsdl:output name="analyticProcessResponse">
            <soap:body use="literal"/>
        </wsdl:output>
    </wsdl:operation>
    <wsdl:operation name="getAvailableProcesses">
        <wsdl:input name="availableProcessesRequest">
            <soap:body use="literal"/>
        </wsdl:input>
        <wsdl:output name="availableProcessesResponse">
            <soap:body use="literal"/>
        </wsdl:output>
    </wsdl:operation>
</wsdl:binding>

```

```

    </wsdl:operation>
  </wsdl:binding>

  <wsdl:service name="AnalyticManagerService">
    <wsdl:port binding="tns:AnalyticManagerSOAPBinding"
      name="AnalyticManagerSOAPBinding">
      <soap:address
        location="http://localhost:8093/analytics/AnalyticManagerService/" />
    </wsdl:port>
  </wsdl:service>
</wsdl:definitions>

```

sampleService1.wsdl:

```

<wsdl:definitions xmlns:wsdl="http://schemas.xmlsoap.org/wsdl/"
  xmlns:sch="http://analyticservices.tut.fi/sampleService"
  xmlns:soap="http://schemas.xmlsoap.org/wsdl/soap/"
  xmlns:tns="http://analyticservices.tut.fi/sampleService"
  xmlns:reqns="http://analyticservices.tut.fi/analyticService/request"
  xmlns:respns="http://analyticservices.tut.fi/analyticService/response"
  targetNamespace="http://analyticservices.tut.fi/sampleService">

  <wsdl:documentation>
    Quality information of sampleService_1:
    [perspective1[price:2.0,accuracy:4.0,time:3.0,availability:6.0];
    perspective2[price:3.0,accuracy:4.0,time:3.5,availability:0.0]]
  </wsdl:documentation>

  <wsdl:types>
    <schema xmlns="http://www.w3.org/2001/XMLSchema"
      xmlns:xs="http://www.w3.org/2001/XMLSchema"
      elementFormDefault="qualified"
      targetNamespace="http://analyticservices.tut.fi/sampleService">
      <xs:import
        namespace="http://analyticservices.tut.fi/analyticService/request"
        schemaLocation="request.xsd">
      </xs:import>
      <xs:import
        namespace="http://analyticservices.tut.fi/analyticService/response"
        schemaLocation="response.xsd">
      </xs:import>
    </schema>
  </wsdl:types>

  <wsdl:message name="analyticServiceRequest">
    <wsdl:part element="reqns:analyticServiceRequest"
      name="analyticServiceRequest"></wsdl:part>
  </wsdl:message>
  <wsdl:message name="analyticServiceResponse">
    <wsdl:part element="respns:analyticServiceResponse"
      name="analyticServiceResponse"></wsdl:part>
  </wsdl:message>

  <wsdl:portType name="SampleService">
    <wsdl:operation name="analyticService">
      <wsdl:input message="tns:analyticServiceRequest"
        name="analyticServiceRequest">
      </wsdl:input>
      <wsdl:output message="tns:analyticServiceResponse"

```

```

    name="analyticServiceResponse"></wsdl:output>
</wsdl:operation>
</wsdl:portType>

<wsdl:binding name="SampleServiceSoap11" type="tns:SampleService">
  <soap:binding style="document"
    transport="http://schemas.xmlsoap.org/soap/http" />
  <wsdl:operation name="analyticService">
    <soap:operation soapAction="" />
    <wsdl:input name="analyticServiceRequest">
      <soap:body use="literal" />
    </wsdl:input>
    <wsdl:output name="analyticServiceResponse">
      <soap:body use="literal" />
    </wsdl:output>
  </wsdl:operation>
</wsdl:binding>

<wsdl:service name="SampleService">
  <wsdl:port binding="tns:SampleServiceSoap11" name="SampleService">
    <soap:address
      location="http://localhost:8190/cxf/AnalyticServiceSampleService_1" />
  </wsdl:port>
</wsdl:service>
</wsdl:definitions>

```

sampleService2.wsdl:

```

<wsdl:definitions xmlns:wsdl="http://schemas.xmlsoap.org/wsdl/"
  xmlns:sch="http://analyticsservices.tut.fi/sampleservice"
  xmlns:soap="http://schemas.xmlsoap.org/wsdl/soap/"
  xmlns:tns="http://analyticsservices.tut.fi/sampleservice"
  xmlns:reqns="http://analyticsservices.tut.fi/analyticService/request"
  xmlns:resps="http://analyticsservices.tut.fi/analyticService/response"
  targetNamespace="http://analyticsservices.tut.fi/sampleservice">

  <wsdl:documentation>
    Quality information of sampleService_2:
    [perspective1[price:4.0,accuracy:5.0]]
  </wsdl:documentation>

  <wsdl:types>
    <schema xmlns="http://www.w3.org/2001/XMLSchema"
      xmlns:xs="http://www.w3.org/2001/XMLSchema"
      elementFormDefault="qualified"
      targetNamespace="http://analyticsservices.tut.fi/sampleservice">
      <xs:import
        namespace="http://analyticsservices.tut.fi/analyticService/request"
        schemaLocation="request.xsd">
      </xs:import>
      <xs:import
        namespace="http://analyticsservices.tut.fi/analyticService/response"
        schemaLocation="response.xsd">
      </xs:import>
    </schema>
  </wsdl:types>

  <wsdl:message name="analyticServiceRequest">

```

```

    <wsdl:part element="reqns:analyticServiceRequest"
      name="analyticServiceRequest"></wsdl:part>
  </wsdl:message>
  <wsdl:message name="analyticServiceResponse">
    <wsdl:part element="respns:analyticServiceResponse"
      name="analyticServiceResponse"></wsdl:part>
  </wsdl:message>

  <wsdl:portType name="SampleService">
    <wsdl:operation name="analyticService">
      <wsdl:input message="tns:analyticServiceRequest"
        name="analyticServiceRequest">
      </wsdl:input>
      <wsdl:output message="tns:analyticServiceResponse"
        name="analyticServiceResponse"></wsdl:output>
    </wsdl:operation>
  </wsdl:portType>

  <wsdl:binding name="SampleServiceSoap11" type="tns:SampleService">
    <soap:binding style="document"
      transport="http://schemas.xmlsoap.org/soap/http" />
    <wsdl:operation name="analyticService">
      <soap:operation soapAction="" />
      <wsdl:input name="analyticServiceRequest">
        <soap:body use="Literal" />
      </wsdl:input>
      <wsdl:output name="analyticServiceResponse">
        <soap:body use="Literal" />
      </wsdl:output>
    </wsdl:operation>
  </wsdl:binding>

  <wsdl:service name="SampleService">
    <wsdl:port binding="tns:SampleServiceSoap11" name="SampleService">
      <soap:address
        location="http://localhost:8190/cxf/AnalyticServiceSampleService_2" />
    </wsdl:port>
  </wsdl:service>
</wsdl:definitions>

```

sampleService3.wsdl:

```

<wsdl:definitions xmlns:wsdl="http://schemas.xmlsoap.org/wsdl/"
  xmlns:sch="http://analyticsservices.tut.fi/sampleservice"
  xmlns:soap="http://schemas.xmlsoap.org/wsdl/soap/"
  xmlns:tns="http://analyticsservices.tut.fi/sampleservice"
  xmlns:reqns="http://analyticsservices.tut.fi/analyticService/request"
  xmlns:respns="http://analyticsservices.tut.fi/analyticService/response"
  targetNamespace="http://analyticsservices.tut.fi/sampleservice">

  <wsdl:documentation>
    Quality information of sampleService_3:
    [perspective1[price:7.0,accuracy:5.0,availability:6.5];
    perspective2[price:6.0,accuracy:4.0,availability:7.0]]
  </wsdl:documentation>

  <wsdl:types>
    <schema xmlns="http://www.w3.org/2001/XMLSchema"

```

```

xmlns:xs="http://www.w3.org/2001/XMLSchema"
elementFormDefault="qualified"
targetNamespace="http://analyticServices.tut.fi/sampleService">
  <xs:import
    namespace="http://analyticServices.tut.fi/analyticService/request"
    schemaLocation="request.xsd">
  </xs:import>
  <xs:import
    namespace="http://analyticServices.tut.fi/analyticService/response"
    schemaLocation="response.xsd">
  </xs:import>
</schema>
</wsdl:types>

<wsdl:message name="analyticServiceRequest">
  <wsdl:part element="reqns:analyticServiceRequest"
    name="analyticServiceRequest"></wsdl:part>
</wsdl:message>
<wsdl:message name="analyticServiceResponse">
  <wsdl:part element="resps:analyticServiceResponse"
    name="analyticServiceResponse"></wsdl:part>
</wsdl:message>

<wsdl:portType name="SampleService">
  <wsdl:operation name="analyticService">
    <wsdl:input message="tns:analyticServiceRequest"
      name="analyticServiceRequest">
    </wsdl:input>
    <wsdl:output message="tns:analyticServiceResponse"
      name="analyticServiceResponse"></wsdl:output>
  </wsdl:operation>
</wsdl:portType>

<wsdl:binding name="SampleServiceSoap11" type="tns:SampleService">
  <soap:binding style="document"
    transport="http://schemas.xmlsoap.org/soap/http" />
  <wsdl:operation name="analyticService">
    <soap:operation soapAction="" />
    <wsdl:input name="analyticServiceRequest">
      <soap:body use="Literal" />
    </wsdl:input>
    <wsdl:output name="analyticServiceResponse">
      <soap:body use="Literal" />
    </wsdl:output>
  </wsdl:operation>
</wsdl:binding>

<wsdl:service name="SampleService">
  <wsdl:port binding="tns:SampleServiceSoap11" name="SampleService">
    <soap:address
      location="http://localhost:8190/cxf/AnalyticServiceSampleService_3" />
  </wsdl:port>
</wsdl:service>
</wsdl:definitions>

```

APPENDIX C: PACKAGE DIAGRAM

This section displays the package diagram of the Analytic Manager OSGi bundle (Figure 44).

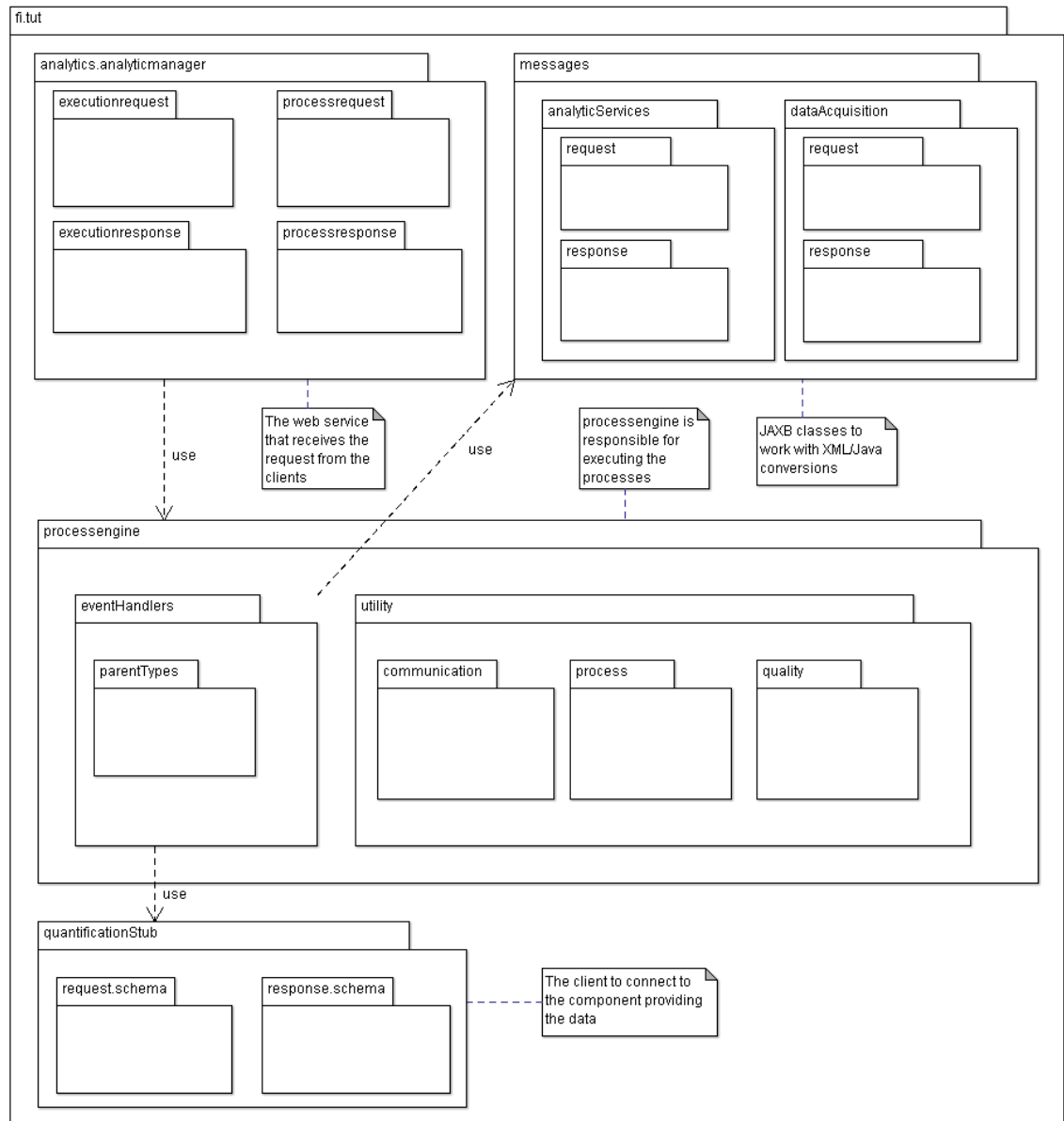


Figure 45: Package diagram of the Analytic Manager component.