Giovanni Di Orio

Licenciado em Engenharia de Automação

Adapter module for Self-Learning Production Systems

Dissertação para obtenção do Grau de Mestre em
Engenharia Electrotécnica, Sistemas e Computadores

Orientador : José António Barata de Oliveira,
Professor Doutor, FCT-UNL

Co-orientador : Gonçalo Moreira Cândido,
R&D Engineer, CTS-UNINOVA

Júri:

Presidente: Prof. Doutor Luís Filipe dos Santos Gomes

Arguente: Prof. Doutor João Paulo Pimentão

Vogais: Prof. Doutor José António Barata de Oliveira
Gonçalo Moreira Cândido

FACULDADE DE
CIÊNCIAS E TECNOLOGIA
UNIVERSIDADE NOVA DE LISBOA

Março, 2013

Adapter module for Self-Learning Production Systems

To my wife and my family

Acknowledgements

I would like to express my gratitude and recognition to Professor José Barata who gave me the opportunity to develop the work presented in this thesis and, above all, to start my research career. His faith in me and in my work were fundamental for making this and future works possible.

A special thank goes to my co-supervisor Gonçalo Cândido who helped me in all the aspects related with this work from the design to the implementation phase, sharing his experience and scientific knowledge as well as friendship. During all the years spent working together, Gonçalo has become one of my most valued friends and I can say with certainty that I am a better person thanks to him.

I would like to thank my colleagues and friends: Pedro Santana, Ricardo Mendonça, Pedro Gomes, Francisco Marques, André Lourenço, Eduardo Pinto, Luis Ribeiro, Magno Guedes and Pedro Deusdado. Our experiments such as the "noise detector" and discussions in lab 1.9 and more recently in lab 1.4 were priceless moments and helped me to become a better engineer.

A special thanks to Carlos Sousa. We spent a lot of time working together firstly during our academic journey and now in research projects.

My deepest gratitude to my family for giving me all the necessary skills to be successful in life. Everything I am today is a reflection of them.

Last but not the least, I would like to thank my wife Mariana for providing me the necessary stability at home, supporting me through my life as student, tolerating my complaints during the most difficult moments and giving me the strength to continue.

Abstract

The dissertation presents the work done under the scope of the NP7 Self-Learning project regarding the design and development of the *Adapter* component as a foundation for the Self-Learning Production Systems (SLPS). This component is responsible to confer additional proprieties to production systems such as lifecycle learning, optimization of process parameters and, above all, adaptation to different production contexts. Therefore, the SLPS will be an evolvable system capable to self-adapt and learn in response to dynamic contextual changes in manufacturing production process in which it operates. The key assumption is that a deeper use of data mining and machine learning techniques to process the huge amount of data generated during the production activities will allow adaptation and enhancement of control and other manufacturing production activities such as energy use optimization and maintenance. In this scenario, the SLPS *Adapter* acts as a doer and is responsible for dynamically adapting the manufacturing production system parameters according to changing manufacturing production contexts and, most important, according to the history of the manufacturing production process acquired during SLPS run time.To do this, a *Learning Module* has been also developed and embedded into the SLPS *Adapter*. The SLPS *Learning Module* represents the processing unit of the SLPS *Adapter* and is responsible to deliver Self-learning capabilities relying on data mining and operator's feedback to up-date the execution of adaptation and context extraction at run time.

The designed and implemented SLPS *Adapter* architecture is assessed and validated into several application scenario provided by three industrial partners to assure industrial relevant self-learning production systems. Experimental results derived by the application of the SLPS prototype into real industrial environment are also presented.

Keywords: Agile manufacturing, Context Awareness, Data mining, SOA

Resumo

Esta dissertação apresenta os trabalhos realizados no âmbito do projecto *NP7 Self-Learning* para o desenho e desenvolvimento de uma arquitectura orientada aos serviços de suporte aos sistemas de produção *Self-Learning* (SLPS). O SLPS será capaz de se auto-adaptar e sobretudo aprender em resposta às alterações no contesto operativo do sistema de manufactura onde ele se encontra a operar. O pressuposto fundamental do trabalho é que uma abordagem sensível ao contesto em conjunto com a aplicação de técnicas de exploração de dados permitirá a adaptação do sistema de produção, o aprimoramento do seu controlo assim como a integração da informação acerca de outras actividades tais quais poupança de energia, manutenção, optimização dos parâmetros de produção e agendamento dinâmico dos recursos. Os componentes fundamentais da arquitectura desenvolvida incluem o *Extractor*, o *Adapter* e o *Learning Module*. O *Extractor* é o componente responsável para observar o processo e coleccionar dados acerca deste para identificar o contexto operativo. O *Adapter* é o componente responsável para adaptar o comportamento do sistema de produção, i.e. os parâmetros do sistema de produção. Finalmente o *Learning Module* é o componente responsável de conferir capacidades de aprendizagem ao *Adapter*. A presente dissertação terá como foco o *Adapter* e o *Learning Module* e apresentará as suas funcionalidades assim como os seus comportamentos explorados durante o funcionamento do SLPS.

Palavras-chave: Manufactura Ágil, Sensitividade ao Contexto, Exploração de Dados, Arquitecturas Orientadas aos Serviços

Contents

List of Figures

List of Tables

Listings

List of Algorithms

List of Notations

$AESOP$	Architecture for Service-Oriented Process - Monitoring and Control
AI	Artificial Intelligence
ANN	Artificial Neural Network
API	Application Programming Interface
BMS	Bionic Manufacturing System
BR	Bosch-Rexroth
CIM	Computer Integrated Manufacturing
$CIMOSA$	Computer Integrated Manufacturing Open System Architecture
CNC	Computer Numerical Control
COM	Common Object Model
$CORBA$	Common Object Request Broker Architecture
$CRISP-DM$	Cross Industry Standard Process for Data Mining
$DBMS$	database management system
$DCOM$	Distributed Common Object Model
DCP	Diagnostic Control Program
DML	Dedicated Manufacturing Lines
DMS	Dedicated Manufacturing System
EAS	Evolvable Assembly System

EPS	Evolvable Production System
ERP	Enterprise Resource Planning
ES	Expert System
EU	European Union
FAS	Flexible Assembly System
FMS	Flexible Manufacturing System
$FP7-NMP$	Seventh Framework Programme - Nanosciences, nanotechnologies, materials and new production technologies
GDP	Gross Domestic Product
$GERAM$	Generalised Enterprise Reference Architecture and Methodology
GUI	Graphical User Interface
HMS	Holonic Manufacturing System
$HTTP$	Hypertext Transfer Protocol
ICT	Information and Communication technologies
$ID3$	Iterative Dichotomiser 3
IDE	Integrated Development Environment
IEC	International Electrotechnical Commission
IMS	Intelligent Manufacturing System
ISA	International Society of Automation
ISO	International Organization for Standardization
IST	Information Society Technologies
IT	Information Technology
$ITEA$	International Test and Evaluation Association
JAR	Java Archive
$JDBC$	Java Database Connectivity
KDD	Knowledge Discovery in Databases
LLD	Ladder Logic Diagram

MAS	Multiagent System
MES	Manufacturing Execution System
OEE	Overall Equipment Effectiveness
OEM	Original Equipment Manufacturer
PLC	Programmable Logic Controller
$R\&D$	Research and Development
RMS	Reconfigurable Manufacturing System
$SIRENA$	Service Infrastructure for Real-time Embedded Networked Applications
$SLPS$	Self-Learning Production System
SOA	Service-Oriented Architecture
$SOAP$	Simple Object Access Protocol
$SOCRADES$	Service Oriented Cross-Layer Infrastructure for Distributed Smart Embedded Devices
$SODA$	Service Oriented Architecture for Devices
SVM	Support Vector Machine
$UDDI$	Universal Description, Discover and Integration
UI	User Interface
UML	Unified Modelling Language
US	United States
$VASS$	Volkswagen, Audi, Seat and Skoda
$WSDL$	Web Services Definition Language
XML	eXtensible Markup Language

1

Introduction

1.1 Motivation

Globalization has created a new and unprecedented landscape changing significantly the way manufacturing companies operate and compete: one of fierce competition, shorter response time to market opportunities and competitor's actions, increased product variations and rapid changes in product demand are only some challenges faced by manufacturing companies of today. As in other domains, production market has deeply felt the effects of globalization on all different layers [Levitt, 1993, Narula, 2003, Noble, 2011]. The increasing demand for new, high quality and highly customized products at low cost and minimum time-to-market delay is radically changing the way production systems are designed and deployed. Success in such turbulent and unpredictable environment requires production systems able to rapidly respond and adapt to changing markets and costumer's needs. To capitalize on the key markets opportunities and winning the competition for markets share, the manufacturing companies are engaged in an innovation race to implement more and more exclusive, efficient and sustainable production systems able to produce innovative and appellative customized products as quickly as possible with reduced costs while preserving product quality.

As stated in [Rao, 2010], manufacturing can be defined as the application of mechanical, physical, and chemical processes to convert the geometry, properties, and/or appearance of a given starting material to make finished parts or products. This effort includes all intermediate processes required for the production and integration of the components of a complex product. The ability to produce this conversion efficiently and effectiveness determines the success of the company. In order to meet these demands, production companies need to optimize their computer controlled manufacturing process parameters finding the best suitable set of parameters for each different production

context. The selection of optimum process parameters represents a fundamental stage to ensure, and eventually increase, manufacturing process efficiency and effectiveness. For such optimization phase, it is imperative taking into account not only production control and execution processes but also associated secondary processes in a fully integrated approach.

Secondary processes, such as maintenance activities, energy saving and/or lifecycle system optimization, have always been very important for industrial production systems. Nevertheless they are typically detached from the core monitoring and control system, implying poor machine tools performance, higher lifecycle production costs and increasing environmental impacts during manufacturing production system operation. As stated in [Jovane et al., 2009], an integrated approach merging the main manufacturing production processes with the named secondary processes will enhance the efficiency and effectiveness of production activities, optimizing fundamental tasks such as maintenance during manufacturing production system lifecycle. Due to the complexity of the problem conventional approaches performing "fail and fix" operation instead of "predict and prevent" practice [Lee et al., 2011] are no longer sufficient. Therefore, a reasonable way to address a fully integrated view is to embed self-learning skills into monitoring and control solutions for manufacturing production system, improving in such a way system capabilities in terms of reconfiguration, monitoring of equipment performance degradation, sustainability. Self-Learning is a new concept in production philosophy where cybernetics principles are applied to manufacturing context to derive more intelligent systems. In the scope of this dissertation, novel technologies such as context awareness and data mining techniques are considered as the foundation for the new generation of manufacturing production systems that will be capable to self-adapt and learn from the continuously changing environment. These capabilities (adaptivity and learning) are guaranteed by a deep use of ontologies and smart algorithms such as Bayesian networks, neural networks, Polynomial Regressions for modelling the manufacturing process from a set of empirical data. As a result, a self-learning production system (SLPS) will be able to both control the production process and constantly monitor all the processes related to it, extract the particular operating context of the manufacturing production system from monitored devices and adapt all the manufacturing process parameters in a holistic and comprehensive way exploiting the representative model of the manufacturing process.

The purpose of this dissertation is to present the research and development related to the proposed SLPS Adapter architecture to integrate a SLPS supported by the use of Service-Oriented Architecture (SOA) principles and technology. The research motivation behind this work relates with the strategic objective of strengthening EU leadership in production technologies in the global marketplace by implementing innovative self-learning solutions to enable tight integration of control and maintenance of production systems. To face this need, the EU FP7-NMP Self-Learning project, focused on the specific needs of the discrete manufacturing industry, involving partners from academia,

research and industry has designed and implemented an highly reliable and service oriented based architecture to support effective self-adaptation of contemporary manufacturing production systems.

1.2 Background: Manufacturing and Society

Manufacturing is the backbone of any industrialized nation representing the cornerstone of any strong economy. The level of manufacturing activities in a country is directly related to its economic health and welfare since manufacturing impulses and stimulates all the other sectors of the society. As a matter of fact manufacturing calls on the skills of everyone from entry-level factory workers to scientists, engineers, and business professionals. In general as [Rao, 2010] has highlighted, an high level of manufacturing activity in a country is a necessary but not sufficient condition to ensure an high standard of living of its people. Manufacturing contributes directly and indirectly for building wealth and generating employment through the whole economy. As stated in [Koren, 2010], finished goods are only a minimal portion of the manufacturing's value. Production of intermediate-level goods also contributes significantly to the economy. The design and production of all manufacturing infrastructure, tooling, equipment and their related software control are industries of their own. Nevertheless, nothing is said about the high levels of transportation, information and communications infrastructure that are all essential to support the whole manufacturing machinery ecosystem. As depicted in figure 1.1, the manufacturing sector was constantly the highest in Gross Domestic Product (GDP) percentage of the developed countries.

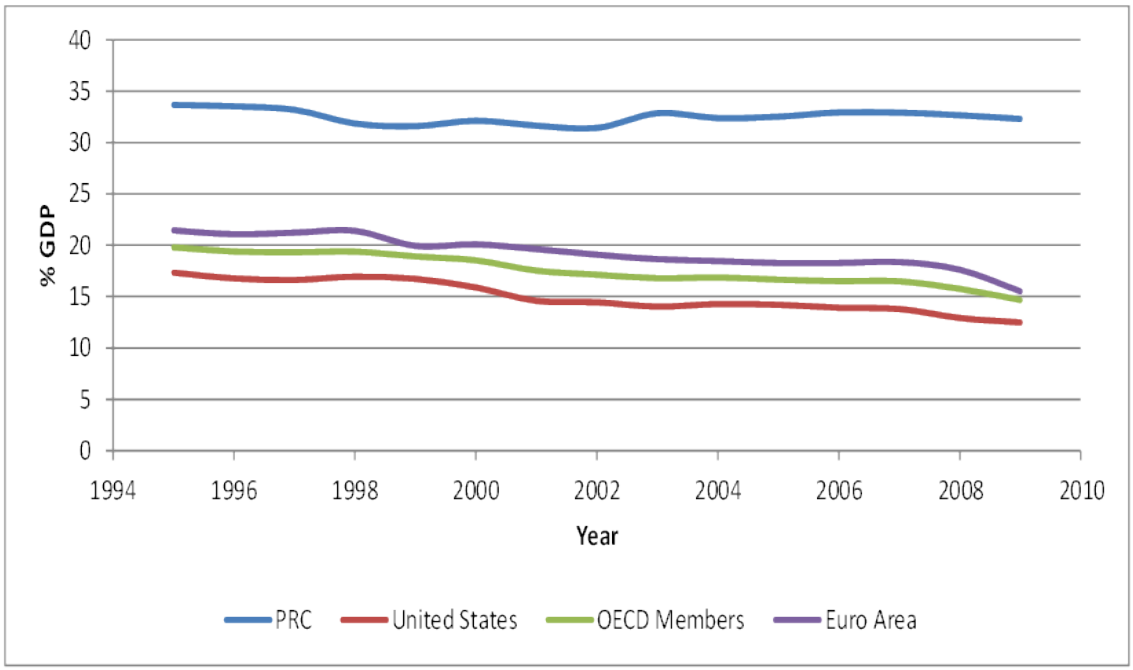


Figure 1.1: Manufacturing contribution to the % of GDP

However the percentage of GDP is declining due to a general outsourcing trend of manufacturing towards low-wage countries allowing development of products more competitive in terms of price. By doing so, according with [Dankbaar, 2007], the outsourced manufacturer has a clear incentive to try to broaden his expertise and to enter into activities for which he was not originally contracted, becoming a competitor of its costumer.

The regional and national governments in the low-wage countries strongly encourage companies to follow that course implying devastating long-term effects on manufacturer contractors and on socio-economic assets. As point out in [Bengtsson and Dabhilkar, 2009], the former refers to the long-term loss of competencies and/or know-how and responsiveness to changes. The latter refers to socio-economic assets reorganization, into developed countries, due to exacerbate compositional changes in workforce that favor highly educated workers penalizing low-skilled workers and increasing unemployment [Morrison Paul and Siegel, 2001].

Counter outsourcing trend introducing new subsidies to support the entire manufacturing structure and investigating for new and more efficient processes supported by technological evolution, represent a necessary condition for enhancing manufacturing and assuring the wealth of a country while improving its competitiveness on a world-scale.

1.3 Research Problem

Nowadays the globalization, the emergent technologies and the requirements for system flexibility, cost reduction and performance improvement lead to the need for robust and agile approaches for self-adaptive manufacturing production systems.

Today trend towards services to support the product lifecycle and the need for more and more product variety imposes a paradigm change in manufacturing production processes from a very static approach to new one that integrates the concept of evolution over time and, at the same time, new eco-freindly/green requirements. Since the type of equipment is usually heavy and bulky which turns change more difficult, the attention will be then focused on the control and supervision system at shop-floor level, representing the main degree of freedom to ensure adaptability still guaranteeing reliability, availability and safety of manufacturing production systems. However, the existence of hard industrial property standards, different for each manufacturer, represents a big obstacle to the introduction of new methods and procedures in automation shop-floor software development, opening the doors to the development of adds-on solutions capable to work in harmony with the existent control and supervision systems while optimizing their activities.

In this scenario, the self-learning project addresses a generic and easy-to-integrate architecture for context awareness and self-adapting based solution to improve existent production systems in terms of manufacturing control and supervision optimization

tasks, integration of secondary processes in main control and final product quality. The self-learning solution aims to work together with the main manufacturing process control system in a cooperative way without affecting correctness and safety of the whole automation process. To do that, the methodology used for the design and development of such architecture is generic enough to allow its applicability and integration inside a wide variety of manufacturing production systems.

The self-learning architecture will be able to "sense" the operating context of the manufacturing production system in which it is embedded, "extract" all the necessary context information, "reason" on it inferring conclusions about the best fit parametrization for the current context and, finally, "adapt" the manufacturing production system according to it or "learn" from human expert decision if some parameters have to be changed. To do this, two main generic components have been developed and implemented, namely: the context *Extractor*, responsible for identifying the current manufacturing process context, and the *Adapter* representing the main focus of this dissertation, which is responsible for exploiting supervised machine learning classification/regression techniques in order to adapt the manufacturing production process parameters for improving production system productivity, efficiency and performance under the extracted context.

Communication between all the modules inside self-learning architecture as well as between the self-learning architecture and the manufacturing process controller is provided by standardized and easy to integrate Information and Communication technologies (ICT) solutions.

The basic concept of a self-learning system is depicted in figure 1.2

The self-learning approach is intended to have a high impact on manufacturing industries and solve open questions concerning:

- Reduction of time, efforts and the possibility of errors during the activity of determining optimal process parameter setting.
- High degree of flexibility in the development and installation of production monitoring and supervision systems.
- Reduction of down times during product exchange and/or conflicts situations.
- Increasing of Overall Equipment Effectiveness (OEE), i.e. plant availability and its productivity over time.

This research initiative was driven by several application scenarios applied to three real world industrial environment: Integration of control and energy efficiency optimization on Computer Numerical Controlled (CNC) production processes, enhancement of flexibility of machines for the shoe production industry, and optimized job dispatching on flexible production cells for the automotive sector. The purpose of the application scenarios is to assure industrial relevance of the self-learning concept and methodology as well as genericity of self-learning solution architecture.

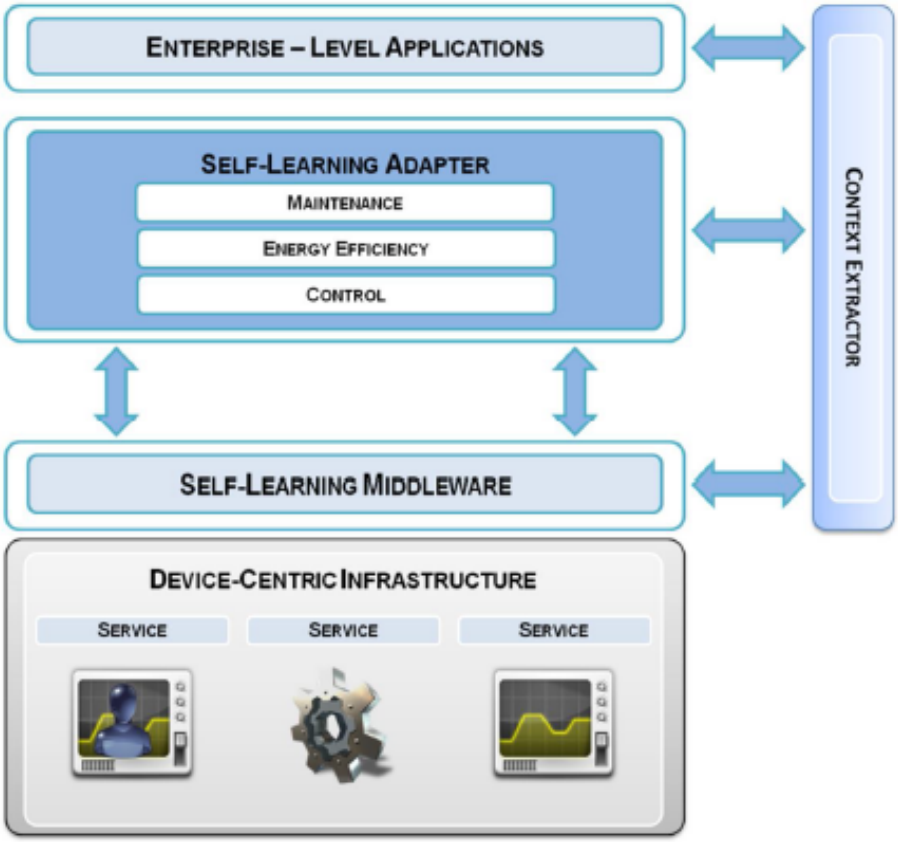


Figure 1.2: Basic concept of a Self-Learning system [Self-Learning, 2010b]

1.4 Approach

As stated in section 1.3, the self-learning project aims to design and implement an architecture capable to confer evolutionary properties to a manufacturing production system through the concepts of: sensing, extracting, reasoning, adapting and learning. Therefore, taking into account that the basic components of the self-learning architecture are the context *Extractor* and the *Adapter*, the sensing and extracting features are provided by the former while the adapting and learning features are provided by the latter.

The objective and contribution of this dissertation is to design and implement the *Adapter* component architecture. This work was the final result of a process constituted by the following steps:

1. **Background Review**: in this step a deep review about todays manufacturing systems limitations, challenges and new requirements has been conducted identifying the main research question: How to design and implement a standardized and easy to integrate platform ensuring adaptation and/or evolution of manufacturing production systems over time? Which is best technological approach to follow?

2. **Problem Characterization and Description**: in this step industrial and research project partners strongly cooperated for identifying possible application scenarios.

The result of this phase was a detailed description of the identified application scenarios as well as extraction of a set of needs and requirements.

3. **Adapter Requirements**: in this step a set of requirements for the *Adapter* component defined according with the main research question and the identified application scenarios.

4. **State-of-the-Art Analysis**: in this step a comprehensive analysis about existent and/or available theoretical approaches, tools and services has been conducted in order to provide the necessary background upon which the Self-Learning solution can be designed and implemented.

5. **Design and Implementation**: in this step the *Adapter* component architecture is designed and a technological solution is implemented according to the requirements defined in the previous steps.

6. **Experiments**: in this step a test-bench has been defined for evaluating the outcomes of the *Adapter* designed architecture. For each test set the *Adapter* outputs has been collected for further analysis.

7. **Validation**: finally, after collecting the system outputs, a validation phase becomes necessary to prove the concept. Results validation has to take into account both the nature of the problem and the results values for corroborating the validity and the correctness of the applied methodology, approach and entire concept.

Finally, the figure 1.3 illustrates the above steps.

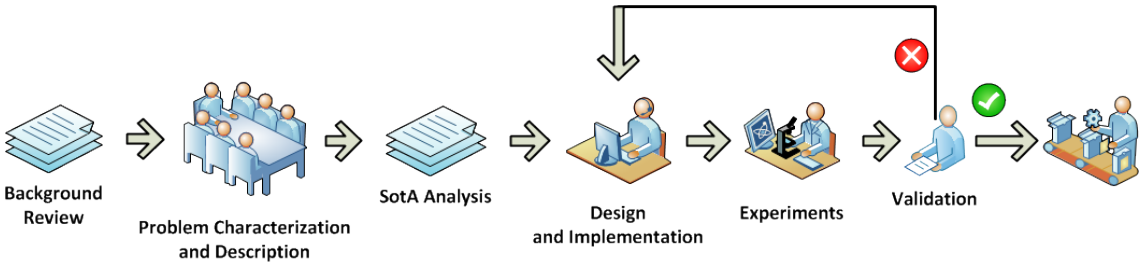


Figure 1.3: Design and Development Approach

1.5 Dissertation Outline

This dissertation is organized as follows:

- Chapter 2: presents a review of the state-of-the-art for manufacturing production systems together with fundamental supporting concept that have driven the design and development of the SLPS prototype.

- Chapter 3: presents an overview on the general architecture for Self-Learning Production System and focuses on the SLPS *Adapter* component that is responsible for ensuring manufacturing production system adaptation and for enhancing control and other manufacturing aspects such as energy use optimization and maintenance relaying on data mining techniques.
- Chapter 4: describes all the works done for implementing, assessing and validating the generic reference architecture for the SLPS *Adapter*.
- Chapter 5: gathers a set of conclusions, main contributions addressed by the dissertation and possible further research scenarios on the subject of SLPS.

2

State-of-the-Art Analysis

2.1 Society, Economy and Technology: Manufacturing Paradigm enablers

From the very beginning, the social, economic and technological aspects imposed a set of manufacturing requirements on manufacturing companies influencing the way their manufacturing processes were organized. Changes in society organization and structure as well as in markets and economy conditions trigger new and more challenging requirements for manufacturing industry determining the death and born of several revolutionary business paradigms.

According to [Koren, 2010], industry has replied to market and societal changes and imperatives by developing new manufacturing processes to produce products, and new manufacturing business paradigms to sell them.

A manufacturing business paradigm or simply business paradigm represents a strategic approach with the objective of creating values for the manufacturing company taking into account three essentials elements, namely: economic value, competitive advantage (over competitors), and value to the costumer. The business model should define who the customer is and how to create economic value for the company by providing customers with a product or service from which they can derive benefit.

Since, business paradigms are basically related with business area, i.e. with marketing and selling a product and/or service, the design and development of new business paradigms may not be enough to ensure manufacturing company competitiveness and its success and prosperity, especially without a global strategy that should take into account product development, manufacturing process together with economic and social issues.

The business paradigms of a manufacturing company must be supported by production capability, confirming that technological aspects play a fundamental role for enabling the development of new manufacturing processes capable to produce new, appellative and quality products at reduced cost and time-to-market while satisfying customer demand. Therefore, the use of competitive and up-to-date technologies is one of the key factors for enabling companies competitiveness in market sharing. However, technological issues alone are not able to increase manufacturing companies productivity, responsiveness and preparedness to market opportunities and competitor's actions without a global strategy that determines which products to develop, for which regions on the globe, where to locate factories, how to integrate global supply chains, and how to boost productivity with the same global resources as exposed in [Koren, 2010]. As a matter of fact sterile investment in technological advance without any global strategy could lead to worst results than before and in the worst case could cause the end of the company itself because of the loss of competitiveness. In this sense, a deep integration between business paradigms, manufacturing processes and product architecture give life to a new manufacturing paradigm.

Since technological aspects, applied to manufacturing processes, represent the major topic of this dissertation, this chapter provides the essential supporting concepts necessary to frame the research activity. It has been divided into two main parts. The first one briefly introduces an historical perspective on the principal manufacturing business paradigms focusing on the manufacturing process organization end its evolution along time, while the second provides literature theoretical approaches, methodologies, solutions and tools and/or more in general scientific and technological activities directions, supporting concepts and areas relevant for the Self-Learning approach.

2.2 Agility as a research vector

The concept of agility covers different areas of manufacturing, from management to the shop-floor. As exposed in [Yusuf et al., 1999], an agile manufacturing enterprise should be capable to detect the rapidly changing needs of the marketplace and propagate these needs to the lower levels of the enterprise in order to shift quickly among products and models or between product lines in response to them. Therefore, it is a top down enterprise wide effort that supports time-to-market attributes of competitiveness [Noaker, 1994]. Thus to be agile, a manufacturing company needs to integrate product and process design, engineering and manufacturing with marketing and sale in a holistic perspective. In this context, ICT represents the cornerstone for achieving agility.

The concept of agility has been widely debated in the literature [Goldman et al., 1995, Kidd, 1995, Oleson, 1998, Goranson, 1999, Christopher, 2000, Guarino and Welty, 2002, Nambiar, 2009] For Goldman, Nagel and Preiss agility is the capability of an enterprise to operate in a competitive environment of continually, and unpredictably, changing in customer opportunities [Goldman et al., 1995]. According to Teece agility

pertains to firm success in turbulent environment, including dynamic capabilities [Teece, 2009], strategic flexibility [Hitt et al., 1998, Ansoff, 2006], and market orientation [Kohli and Jaworski, 1990, Narver and Slater, 1990]. Although several definitions for agility have been created, the common stream is on being able to compete and prosper within a state of dynamic and/or continuous change. This involves two main aspects, as stated in [Zhang and Sharifi, 2007], namely:

1. **Responding effectively to changes**;

2. and **exploiting changes by taking advantage of changes as opportunities**.

Although the notion of agility as a competitive concept has been around for some time, it remains more a concept then a reality. According to [Zhang and Sharifi, 2007] the problem is rooted in the fact that there is a lack of theory and the question of how implement agility in manufacturing company has not been yet properly answered.

Furthermore, Alexopoulou [Alexopoulou et al., 2009] states that there is a great confusion about the relation between flexibility and agility. Some researchers consider them synonyms or almost synonyms [Kidd, 1995, James-Moore, 1996, De Leeuw and Volberda, 1996]. Others specify a dependency relation between them. Evgeniou [Evgeniou, 2002], for instance, considers flexibility as a necessary condition for adaptation and in turn agility. Dove [Dove, 2005] on the other hand, differentiates flexibility from agility by stating that the former refers to the ability to respond to expected changes while the latter concerns unforeseen changes as well.

In the context of this dissertation, the term agile is adopted and considered as the ability to sense environmental changes, extract all the necessary information from the "sensed" changes, start a reasoning process to adapt the manufacturing process parameters according to the extracted information and finally learn from human expert input. The process sense-extract-reason-adapt-learn will take into account the system aims and capabilities together with the entire system life-cycle in order to discover the best suitable parametrization for getting profit from detected changes. Therefore, agility is not a simple response and/or reactive process triggered by environmental changes but incorporates, above all, the ability to proactively cause changes in process to better respond to actual contextual conditions implying self-organization and self-learning features.

Therefore, flexibility means the preparation of manufacturing process for future expected transformations offering a set of possible choices, while agility involves adjustments that a manufacturing process may require as a consequence of unforeseen and unpredictable changes. Agility is then different from flexibility since it embodies the concept of evolution along time even if flexibility is a necessary condition for agility.

2.3 Trends in manufacturing systems

From the nineteenth century to present date manufacturing industry has undergone several revolutionary business paradigms spread over three different ages, namely: industrial age, information age, and Post-information age as depicted in figure 2.1.

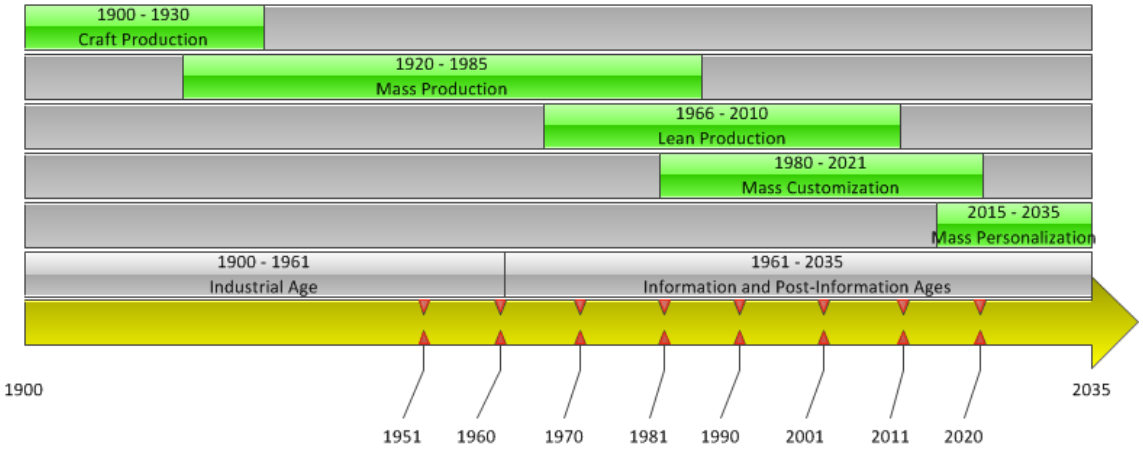


Figure 2.1: Manufacturing business paradigms to present day adapted from [Barata, 2005]

The born and death of each business paradigm as well as the start and end of each age is unclear, i.e. it is impossible to identify exacts points in time where one age and/or business paradigm can be considered terminated and the other starts. On the contrary, during the transition between them it is typical to assist to an overlapping phenomenon where the two paradigms and/or ages can coexist for a long time.

A detailed and very comprehensive overview on manufacturing paradigms, their evolution along time as well as a whole analysis about socio-economic and technological aspects can be found in [Barata, 2005, Ribeiro and Barata, 2011, Koren, 2010].

2.3.1 Industrial Age: craft and mass production

In the late eighteenth and early nineteenth centuries, major changes in manufacturing took place in most of the developed countries. In this period, the industrial age is considered to have started with the advent of the industrial revolution [De Masi, 2000], that triggered a paradigm shift from manual-labor-based economy towards machine based manufacturing or, in other words, from craft production to mass production.

Craft production is the manufacturing paradigm of the end of the nineteenth century, which was mainly mastered and dominated by Europe. In this period, highly skilled workers (Craftsmen), using general-purpose and highly-flexible machines, created exactly the products that the customer asked for, on demand and one item at time. The craft production business paradigm is a pull-type paradigm where a product is purchased by a customer, designed and finally produced. The produced products were not standardized, transforming the production process into "one of a kind" process where unique

products were produced implying high prices with the consequence that only few customers could afford them.

Although craft companies could produce a great variety of products (product flexibility), this was done at a very slow pace. Changing or adapting the manufacturing process to new requirements in terms of product changes or quantities produced was very difficult. Therefore, craft industries, although flexible, were not agile, i.e. they guarantee the production of great variety of product but show big difficulties to quickly adapt their shop-floor when variation in markets (increasing in demand) or changes in product occur.

Finally, the craft production paradigm is characterized by:

- High product flexibility
- Very low volume per product
- Pull-type business paradigm
- Highly-flexible tools
- Skilled work force

In this context the advent of certain conditions such as changes in socio-economic conjecture, mass availability of raw materials and technological development supported, in a deterministic way, the paradigm shift from craft production to mass production.

Mass production means producing extremely large quantities of identical/non diversified products at reduced price. This manufacturing paradigm is normally attributed to Henry Ford that stated and proved that producing in a continuous and synchronized flow assembly line, using interchangeable standard parts, whereby workers are assigned to specific and systematized tasks would increase the global product quality while producing faster and allowing significant drop in the price [Ford and Crowther, 1988, Gross et al., 1996]. The moving production line consists of specialized and very dedicated machines and/or equipments used to assemble, transport and finish products while maintaining high production volumes to produce the same product without variation was the emblem of this period. Because of the extremely large involved quantities, production fixed costs can be spread over these quantities enabling a reduction in the final sale price.

In contrast to craft manufacturing paradigm, the mass production business paradigm is of a push-type where the product is designed by manufacturers and produced asserting that there will always be a possible costumer willing to buy it. The manufacturing processes were then able to produce great quantities of a single product in a very efficient way. Dedicated Manufacturing Lines (DML) were a characteristic of this paradigm; designed and built for ignoring flexibility, agility and adaptivity requirements to privilege efficiency and robustness during the entire manufacturing process.

The dominant characteristics of the mass production paradigm are:

- Null or very limited product variety
- High production volumes
- Push-type business paradigm
- Dedicated machinery and moving assembly lines
- Relatively unskilled work force
- Rigid manufacturing production process

2.3.2 Information and Post-Information ages

The information age started with the dissemination of communication means (based on electronics and computers) and computers among the masses of people, which is considered to have happened in the seventies of the last century. For some authors [Hames, 1994, Bell, 1962, Kornhauser, 1959] the mass society, which is a consequence of a mature mass production, was the most important cause for the transition from the industrial age to the information age. It was believed that mass production/consumption would boost mankind to unprecedented development and sophistication. This would not be verified due to several reasons including social and economic: technological advances and unscrupulous greed for profit increased unemployment; the work organization led to a certain amount of alienation of individuals that worked in factories, often in mindless and repetitive jobs [Toffler, 1980]; the environmental, health and safety costs were high and the progressive and general increase in customer welfare made them increasingly demanding for more sophisticated and customized goods. Product quality and customization arise as a fundamental factor of choice.

The age of uniformity, standardization and "static perfection" gave way to a new turbulent age where variety, customization, better quality, continuous innovation and strong competition in market sharing are the cornerstones. Although there is no consensus about why the industrial age came to a crisis, there is some consensus that the dissemination of electronics and computers are the basis for the change from the industrial age to the information age, mainly because computers and electronic equipment became so disseminated that improved means of communication and organization became possible.

The disruptive factors that led to this age are both socio-economic (mass society saturation) and technological (computer and electronic) underlining the fact that no technological or socio-economic determinism exists, furthermore both factors mutually influence each other's.

Computer technology can therefore be seen as the principal catalyst of a social and economic movement. People are now more sophisticated, aware of their own rights and therefore more and more demanding.

The turbulence and fragmentation of markets are characteristics of this age inducing new and completely different requirements for the production systems, implying important consequences to the way manufacturing processes were designed, implemented and deployed.

The global dissemination of the World Wide Web in this age changes the way people and manufacturing companies interact opening the doors to the Post-Information age where everything is made to order, information is extremely personalized [Negroponte, 1996, Dewan et al., 2000, Tapscott, 2009] and ideally accessible everywhere and to everyone.

2.3.2.1 Lean Production

The lean production means the set of practices evolved to facilitate the lean manufacturing ideal: design and development of an efficient and effective manufacturing production system. Lean manufacturing defines a philosophical and/or a new way of thinking that directly reflects cultural and social-economic dimensions of Japanese society. Although, this paradigm is contemporary to the mass production paradigm its merit is to define operations management methods rely not only on reduced costs but and above all on enhanced product quality ensuring its survival and validity to date.

The concept of lean manufacturing emerged as reaction of the oil crisis and the significant socio-economic changes in the 1950s-70s, but was only with Toyota and its factory organization approach (Toyota Production System) that this paradigm assumes its best and spectacular expression. The term "Lean" manufacturing has been first coined by John Krafick in his article, "Triumph of the Lean Production System" in 1988. Kraficik's research was continued by the International Motor Vehicle Program (IMVP), which produced the book "The Machine that changed the world" revealing to the western world the Japanese production and organizational techniques. A lean manufacturing system is one that meets high throughput or service demands with very little inventory and with minimal waste. The most important idea behind lean manufacturing paradigm is avoiding waste, *Muda*, which is the Japanese word for waste, representing every activity that absorbs resources but creates no value improving, in such a way, product quality while reducing time to market and costs. Therefore, the main difference in respect of the mass production paradigm is the less of everything (i.e. tool investment, manufacturing space, human resources, etc.) focusing on the achievement of the following goals such as continually declining costs, zero defects, zero inventories (item is produced only when it is needed) and endless product variety [Womack et al., 1990].

Another aspect of lean manufacturing is the way the production line (shop floor) is organized. Shop floor workers are organized into teams with a team leader rather than a foreman, as occurred in mass production. The workers are polyvalent and able to execute the various tasks assigned to the team. This provides generally a greater sense of fulfilling in the workers since they are not confined to the repetitive execution of the same tasks

as in mass production. Further, teams have the right to stop the assembly line, whenever they think it is necessary, as when repairing it. Workers are stimulated to participate with suggestions to improve the process. The continuous improvement strategy took place in collaboration with industrial engineers and can be effective because workers, if properly motivated, can contribute substantially since they are the ones that truly master the processes being taken care of [Ribeiro and Barata, 2011].

Although lean supporters claim that lean production system is universally applicable, most of the successful implementations of lean production have been in the automotive and electronic sectors, characterized by high volumes with low products variety. Since endless product variety represents a landmark for lean manufacturing paradigm, the control and supervision system architecture needed to be flexible enough for allowing a great product variety. In addition, next to the product variety another important aspect is represented by the continuous improvement implying new requirements for system changes and adaptations along shop-floor life cycle (agility). However, although flexibility and agility are both necessary, the control and supervision system architecture was designed looking for mainly flexibility rather than agility. Shop floor reengineering process was then slow and difficult to perform.

2.3.2.2 Mass Customization

Mass customization is a society-driven paradigm that started in the 1980s. As the market for a product matures and the customer become wealthier, they are not anymore satisfied only with low costs product but begin to look for a large variety of products to choose from, in order to have their preferences fulfilled at the same low cost as in mass production.

In response to the changes in society and consequently in market diversification, manufacturers start to offer no more a "simple" standard product but a new "complex" one, comprising the standard product and a set of extra features and/or packages that, in turn, are offered to customers to configure their own products.

In the mass customization paradigm, the manufacturers decide on the basic product options they can practically offer and produce in large volumes using mass production techniques, while customers select the package that they prefer, buy it, and only then the product is finished. The association of personalized packages to the products is becoming more and more important as confirmed in the automotive sector where this association is already a trend [Juehling et al., 2010].

Differently form the craft production paradigm, the mass customization does not mean producing one-of-a-kind products but developing multiple sets of practical variation of a standard product that can be produced on a mass production system and offered to potential customers. The identification and fulfillment of the "wants and needs" of individual customers without sacrificing efficiency, effectiveness and low costs in production activities are the basic requirements and challenges. The major objective is take

advantage from mass production paradigm assets benefiting from low production costs while adding configuration to the final assembly.

Mass customization paradigm helps to understand why products life cycles are decreasing, why manufacturing processes need to evolve along time and why networked organizations are emerging. To be able to face the new context, manufacturing companies need to change the way to design production processes and products. The production processes should be as decoupled as possible from the ever-changing flow of products for minimizing any machines set-up time, due to product alteration, and still maintaining low production costs. The products design should be based on the concept of modularity and linked to the manufacturing production process allowing the final customized product to be assembled from a set of mass produced components.

The key enablers of mass customization are:

- **Innovation in process technologies** results in the birth of Flexible Manufacturing Systems (FMSs) and/or Flexible Assembly Systems (FASs) which make extensive use of CNC machine tools and computer-controlled handling as well as welding and assembly robots that can quickly switchover production from one product type to another.

- **Marketing networks and customer-plant direct communications**: the mass customization paradigm started with aggressive direct marketing. The Internet allows customer's orders that are directly communicated to the manufacturing plant, enabling more efficient planning strategies and resources scheduling.

- **Socio-economic conjuncture** revealing the willingness for new, completely personalized and unique products.

Therefore, requirement for more customized products implies that manufacturing processes should be designed in order to handle high volumes of product with high mix for satisfying all kind of customers.

The continuous trend for more and more customized and/or personalized products is pushing the customer more and more inside the whole manufacturing process, i.e. directly and actively involved in the design of the product that he wants to buy.

Furthermore, design activity will not be just manufacturing company concern but, on the contrary, will be splitted between the manufacturing company which is responsible for the product architecture, basic modules and their interfaces, and the customer which, in turn, creates his own product by selecting and composing the available modules.

2.3.2.3 Mass customization: types of Manufacturing process

Each manufacturing paradigm is characterized by new and, above all, always different challenges that have to be faced. The design and implementation of State-of-the-Art manufacturing processes, supported by new technologies available at the time in which manufacturing paradigm started, arise as primary response to address the emerging challenges. Therefore, technological development defines the feasibility of requirements imposed by socio-economic conditions.

Several manufacturing processes, based on the most diverse technologies, architectures, approaches and methodologies, has been designed and implemented through the years to satisfy mass customization requirements while improving and/or ensuring manufacturing company competitiveness and position in market sharing.

The most popular key words found in literature for defining different trends in response to paradigm requirements are: flexible, reconfigurable, lean, holonic, bionic, evolvable, agile, self-adaptive and self-learning. All these trends represent different degrees and modes to provide/incorporate agility inside manufacturing processes.

2.3.2.3.1 Flexible Manufacturing/Assembly System (FMS/FAS)

As the customers started looking for new innovative, high quality, low cost and personalized products increasing market fragmentation, manufacturing companies first concentrated on massive investments in technology, namely on automation and software to manage their manufacturing production processes for better respond to markets demand.

However, the use and dissemination of heterogeneous computer hardware and software in manufacturing companies, without a global strategy, corroborating the must for integration between all manufacturing systems subparts.

The Computer Integrated Manufacturing (CIM) paradigm [Browne et al., 1988, Ranky, 1990, Scheer, 1991, Mitchell, 1991, Waldner, 1992, Camarinha Matos et al., 1995], aiming to create a global architecture for modeling and accommodating all the different tasks in a company and, simultaneously, providing and integrated view of manufacturing company, became an import effort towards the goal of increasing the competitiveness of manufacturing companies through the introduction of automation and wider use of computers.

The advent of CIM provided a reference architecture for designing and implementing manufacturing processes introducing, at same time, a certain degree of flexibility.

Flexibility in manufacturing production systems represents the ability to adapt the manufacturing process to produce a range of diversified but mostly predetermined products. Manufacturing companies realized that flexibility was the key to cope with an increased tendency towards more diversified products, with varying demand and with market fragmentation. The production of great volumes of the same product had been replaced with the production of low/moderate volumes with high mix. Manufacturing

engineers, in the effort to create systems able to cope with varying products and demands, developed approaches like FAS and FMS.

The introduction of flexibility into manufacturing production process meant the design and implementation of machines able to perform a wide range or ideally all kind of tasks with the same performance. High investments in technology, introduction, in automation context, of the new software capabilities and the widespread use of computer network for enabling communication inside all levels of manufacturing companies (management, production, shop-floor, etc.) were thought as the basis for manufacturing companies to succeed. However, large investment in technology only leads to sophisticated and expensive machines, with a lot of capabilities, that in some cases are not fully exploited and in the worst case are unused or completely useless for the context of application. Moreover, manufacturing processes are flexible regarding the predetermined products but inflexible regarding the introduction of new products due to the complexity of automatically making the required adjustments [Leitão, 2004].

2.3.2.3.2 Reconfigurable Manufacturing System (RMS)

The Reconfigurable Manufacturing Systems (RMSs) [Koren et al., 1999, Mehrabi et al., 2002, ElMaraghy, 2005] are manufacturing systems that bridge the gap between the Dedicated Manufacturing Systems (DMSs) and the Flexible Manufacturing Systems (FMSs).

The DMS is a manufacturing production system designed to produce very large quantity of the same product. Since the production is dedicated to only one product, the system is robustly designed and optimized to promote productivity in a very efficient way rather than adaptability, configuration and evolution along time.

On the contrary, FMS is a manufacturing system designed to face market conditions and customer demand in terms of product variety focusing on a profound customization. It is constituted by several complex machines that are capable of performing a variety of operations, and by extension can produce a large range of different products penalizing the product volumes, i.e. the relationship between product volumes and flexibility is inversely proportional.

A Reconfigurable Manufacturing System put together both DMS and FMS benefits since it is focused on producing a quite large spectrum of product, showing smaller flexibility than a FMS, while keeping the throughput of a DML. In such a way a RMS tries to gather all the strengths of a FMS and DMS reducing and/or at least eliminating their drawbacks. The key feature of RMS is that, unlike DML and FMS, its capacity and functionality are not fixed. Since a RMS is designed to "reconfigure", to grow and to evolve during its life-cycle, it can respond quickly to the markets changes.

Although RMS and FMS acting on the same environment, they are different approaches or answer to face with new socio-economic challenges; the latter sustains the deep exploitation of highly specialized machines providing generalized flexibility for anticipated

variations of a product, while the former sustains the need for open advanced reconfigurable control and dedicated inter-modular tools promising customized flexibility on demand in a short time and cost-effective. In other words a RMS is designed to provide agility enabling changes in production capacity and in its functionality without affecting its overall robustness and/or reliability. Thus, the key dimensions of RMS are in this context: modularity, integrability, flexibility, scalability, convertibility and diagnosability.

2.3.2.3.3 Holonic Manufacturing System

The holonic manufacturing paradigm was developed in the framework of the Intelligent Manufacturing Systems (IMS) programme. It is inspired by Arthur Koestler that in his work [Koestler, 1968] proposed, for the first time, the word *holon* as a basic unit of organization in biological and social systems. It is the combination of the Greek word *holos* that means "whole", with the suffix *-on* that suggests a particle and/or part. As stated in [Van Brussel et al., 1998], the Holonic Manufacturing System (HMS) paradigm transposes the concepts that Koestler developed for social organization and living organisms into the manufacturing production system world.

The following list of definitions [Christensen, 1994], already developed by the HMS consortium, is essential for understanding the key holonic concepts applied to manufacturing system:

- **Holon**: An autonomous and cooperative building block of a manufacturing system for transforming, transporting, storing and/or validating information and physical objects. A holon can be a combination of other holons and be itself a part of another holon, confirming the idea that a holon is a whole and a part at the same time;
- **Holarchy**: a system of holons which can cooperate to achieve a goal or an objective, defining the basic rule for cooperation;
- and **Holonic Manufacturing System**: a holigarchy which integrates the entire range of manufacturing activities from order booking through design, production and marketing to realize the agile manufacturing enterprise.

The goal of a HMS is to bring in manufacturing context the benefits that holonic organizations provide to living organisms and societies, i.e. stability in the face of disturbances, adaptability and flexibility while preserving efficient use of all the resources of the manufacturing system. Therefore, according with today's manufacturing systems requirements for high adaptability and flexibility, a Holonic manufacturing aims at developing reconfigurable, scalable, flexible, and responsive manufacturing systems. An extensive review about HMS can be found in [Babiceanu and Chen, 2006].

2.3.2.3.4 Bionic Manufacturing System

The Bionic Manufacturing System (BMS) concept and paradigm was first introduced by Okino [Okino, 1993]. This approach relies on the observation of natural systems for

developing manufacturing systems. The structure of natural life exhibit autonomous and spontaneous behavior, and social harmony within hierarchically ordered relationship [Tharumarajah, 1996]. The BMS paradigm is inspired in the functioning of natural organs. Organs of a life-form seemingly act on their own while coordinating their actions and maintaining their harmony. Organs, in turn, consist of components such as cells, and support life forms of which they are a part [Tharumarajah, 1996]. This approach underlines the idea of a hierarchical system where information travels bottom up and top down along chain.

2.3.2.3.5 Evolvable Manufacturing Systems: EAS/EPS

In recent years, a new and innovative manufacturing paradigm emerged for achieving agility in assembly and production systems, where evolution represents its principal keyword. Several research works by many different authors have been conducted on this subject [Onori, 2002, Shen et al., 2006, Barata et al., 2006, Frei et al., 2007, Neves and Barata, 2009].

The Evolvable Assembly/Production System (EAS/EPS) paradigm proposes a solution which, being based on many simple, re-configurable, task-specific elements (systems modules), enables for a continuous evolution of the assembly/production system [Shen et al., 2006].

Therefore, EAS/EPS approach focus on targeting agility through modularity and stepwise evolution. An EAS/EPS is, basically, a system that can dynamically adapt itself to new products and production scenarios allowing evolution of the system together with the environment, i.e. addition and removal of manufacturing modules in response to changes in production orders and plans at run-time, without the need to completely stop the system for programming tasks.

Since modules (physical components of the system) are the building blocks of EAS/EPS, they should be created in a well-investigated manner providing the necessary level of granularity, representing the lower level of device considered within the reference architecture, and standardized interfaces allowing the addition and/or removal of the modules inside the system architecture (plugability) while ensuring communications between them.

2.3.2.3.6 Self-Learning Production System

The previous paradigms set the theoretical background for what is expected to be the next generation of manufacturing systems. However, an important barrier to prototype implementations of these concepts and principles exists. Traditional control approaches and/or paradigms do not provide all the capabilities needed for implementing agile manufacturing. The existence of rigid and different standards developed by Original Equipment Manufacturers (OEMs) allows on the one hand integration and cooperation between different devices from different suppliers (sensors, actuators, controllers, robots

and so on) resolving any problem that involves interoperability while keeping easy control software debugging and improving its quality and reliability, but on the other hand hinders the introduction of new methodologies and procedures in manufacturing process software development.

In this scenario, the Self-Learning paradigm is intended to deliver additional productivity gains to a manufacturing process extending the reach of the automation system beyond the world of process control considering maintenance and new green requirements without disturbing existing control approaches.

Within the framework of manufacturing production systems, Self-Learning concept proposes application of machine learning techniques to allow computer based control systems to change their own behavior based on the information or, more in general, on the knowledge and patterns extracted from all the available data.

Therefore, a Self-Learning production system combines traditional computer based control systems with the capability to learn from available data along its entire lifecycle. These characteristics enable the fully integration between control and so called secondary processes together with the selection of optimum manufacturing process parameters based not only on the available data but above all on the entire knowledge gathered during system operation. Hence, the Self-Learning concept and methodology confers agility to a manufacturing process by including new components into the overall control system architecture delivering productivity gains, improved product quality and reduced energy consumption thanks to a more intelligent way to use available manufacturing process data boosting automation to a new level far beyond process control.

Since application of machine learning solutions and/or data mining techniques to manufacturing production systems is a relatively new subject of research, it is fundamental to absorb the knowledge either theoretical or particular to other domains already available and sift how to apply these techniques in order to provide an important contribution to future production systems.

2.4 Supporting Concepts and Technologies

The purpose of this section is to provide the necessary concepts on which the proposed solution is founded, allowing a better comprehension of the approach and technologies used to implement this work.

However, the description of the supporting concepts given in this section is not extensive, on the contrary, it is only a small introduction necessary to frame the current work suggesting that the references scattered throughout the text should be consulted for better insight on subjects.

2.4.1 Expert Systems

Expert Systems (ESs) are a branch of applied Artificial Intelligence (AI). As focused in [Liao, 2005], the basic idea behind the ES approach is to gather the vast body of task-specific knowledge of a human expert and transfer it from a human to a computer system. This amount of knowledge is stored inside the system and is used whenever a user call the computer for specific advices i.e. whenever a user needs for help in the process of decision support and problem solving. Therefore, an expert system can be defined as a computer system that behaves like a human expert giving advices and eventually explaining the logic and/or the relevant steps behind the advice [Turban et al., 2005]. As stated in [Rao, 2010], the general architecture of an ES is composed by a knowledge base (obtained by a modeling activity applied on the human expert knowledge) and an inference motor. The knowledge base contains domain knowledge which may be expressed as a combination of "IF-THEN" rules, factual statements, frames, objects, procedures and cases. The inference motor is the part of the ES that manipulates the knowledge base allowing computer system to apply inference mechanism on data for generating advices. Once advices are available, a Graphical User Interface (GUI) will be responsible for showing the result to the system user. Relevant result will be stored into repositories and can be used in future to enhance computer system advices. The ES theory represents a fundamental background for the development of the Self-Learning solutions since some basic concepts are shared such as support to human expert decision making process as well as evolution along time thanks to new knowledge introduced into the system by expert user.

2.4.2 Machine Learning

Several relations can be observed between animal and machine learning and/or way to learn, since many techniques of machine learning are inspired by the efforts of psychologists to make their theories more precise through computer models. Moreover, it seems likely also that the concepts and techniques being explored by researchers in machine learning scope may enable a better understanding of certain aspects of biological learning [Nilsson, 1996].

The word "learning" has many different meanings. According to dictionary the term "To learn" is used, at least, to describe:

- To memorize something;
- To receive instruction;
- To be informed of, ascertain;
- To become aware by information or from observation and exploration;
- To develop a motor and/or cognitive skills through practice;

- and to organize new knowledge into general, effective representations.

However these meanings have some shortcomings when talking about computers. According to [Witten and Frank, 2005] the first two are too passive and trivial for a computer system. Furthermore, a computer system can memorize or be informed of something without being able to apply new knowledge to new situations. The last three meanings reveal the impossibility to test whether learning has been achieved or not.

In the context of this work the definition of "learning" and/or "to learn" provided by Herbert Simon [Simon, 1980] has been followed:

"Learning denotes changes in the system that are adaptive in the sense that they enable the system to do the task or tasks drawn from the same population more efficiently and more effectively the next time."

This definition ties learning to performance rather than knowledge and underlines the idea that learning systems have to acquire some information from examples problem in order to perform better because of it, taking a fundamental advantage of its knowledge. Using a more formal definition provided by [Mitchell, 1997]:

"A computer program is said to "learn" from experience E with respect to some task T and some performance measure P, if its performance on T, as measured by P, improves with experience E".

Therefore, learning implies thinking above acquired data of a problem with the purpose of improving the system behavior.

Machine learning is programming computers to optimize a performance criterion using example data or past experience [Alpaydin, 2004]. Hence, it can be said that a machine learns whenever no directly or explicitly written computer program exists to solve a given problem but, on the contrary, the machine changes its structure, program, or behavior based on inputs or in response to external information in such a manner that its overall performance is expected to improve [Alpaydin, 2004, Nilsson, 1996, Michie et al., 1994, Michalski et al., 1983].

Machine learning is a branch of AI that, in turn, is concerned with intelligent behavior in artifacts. According to [Blum, 2007], machine learning aims to understand the fundamental principles of learning as a computational process, identifying at a precise mathematical level what capabilities and information are needed to learn different kind of tasks successfully, and understanding the basic algorithmic principles involved in getting computers to learn from data and to improve performance with feedback. The typical architecture of a learning system, introduced by Nilsson in [Nilsson, 1996], is represented in

figure 2.2

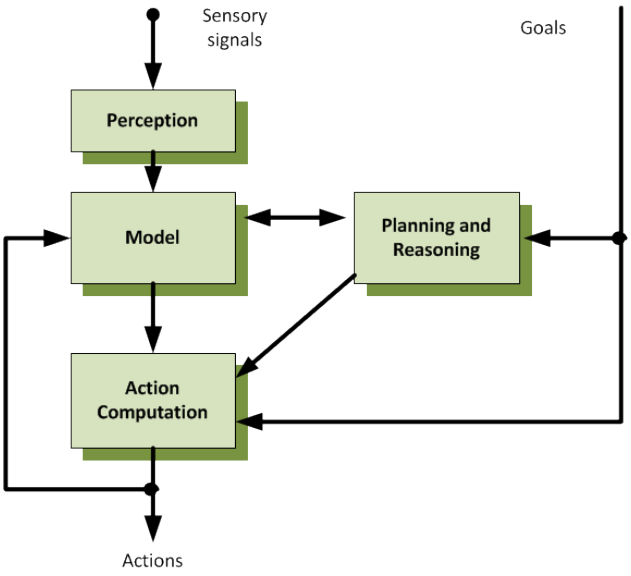


Figure 2.2: Learning system general architecture [Nilsson, 1996]

A learning system is then able to perceive the environment in which it is immersed, create a mathematical model to represent it and based on this and on its own goals reason on this representation for computing appropriate actions, perhaps by anticipating their effects.

The main goal of the Self-Learning project is the development of a learning system transposing the general architecture, depicted in figure 2.2, into the world of manufacturing production system enhancing the capabilities of actual control and monitoring software systems in the context of:

- Tasks that cannot be well defined except by examples or, in other words, when only input/output pairs are available;
- processing large amount of data (too big to be efficiently analyzed by a human) for extracting relationship and correlations between them (data mining) in order to construct a model that can be used to predict future situations with high accuracy;
- Evolvable and/or constantly changing environments such as manufacturing production processes where machine learning techniques can be used to reduce drastically the need for constantly re-engineering as well as optimization tasks over time.

2.4.2.1 Data Mining: Learning from data

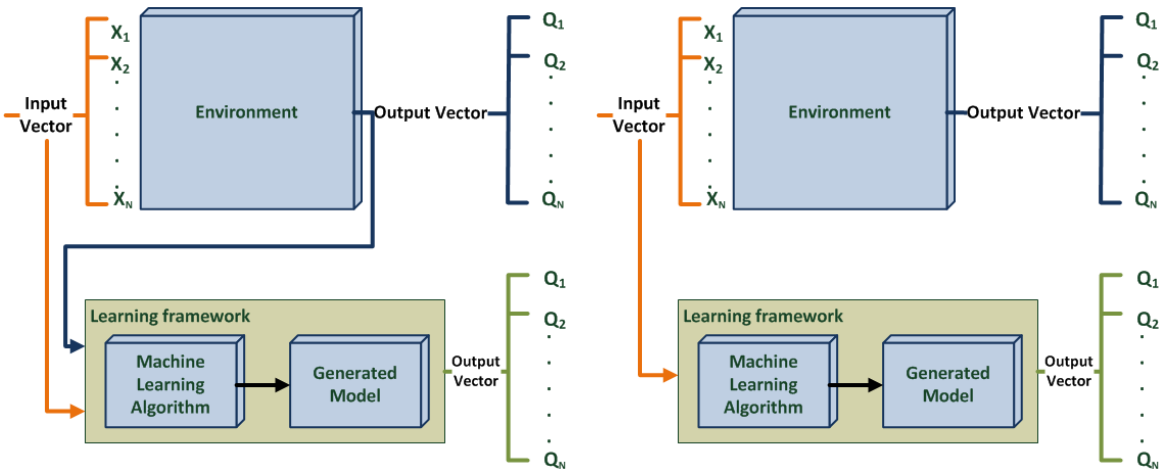
Machine learning is concerned with the development of algorithms that take as input empirical data collected from sensors and/or stored into databases, and yield patterns or predictions that rely on the acquired data.

The process of extracting information or better knowledge from large quantities of data is also known as *data mining* [Fayyad et al., 1996, Witten and Frank, 2005]. Therefore, machine learning provides the technical basis of *data mining* since it furnishes tools (algorithms) able for extracting regularities from data. Different types of algorithms have been developed along time with different characteristics according to the nature of the problem. The table 2.1 shows the different types of existing learning algorithms as well as their applications contexts. According to the taxonomy represented in table 2.1, two major types of learning exist namely the *supervised learning* and the *unsupervised learning* while the others are mostly a generalization of the previous two. However, they are discussed and analyzed as independent research domains.

The *supervised learning* consists in finding the function f that best defines the relation between inputs and outputs by looking input-output training examples m. The main goal is to build a concise model of the problem (represented by the f function), through the learning of a set of rules from the training set, for classifying and/or predicting new unknown/unseen situations (inputs) [Kotsiantis et al., 2007].

On the contrary, the *unsupervised learning* refers to the problem of finding regularities and/or patterns in training example set without any feedback from the environment. The main goal is to build a statistical representation of the input data that will be used for decision making, i.e. classification and regression tasks when no feedback from the environment are available.

Both, *supervised* and *unsupervised learning* approaches, rely with the utilization of machine learning algorithm for identifying a representative model of a problem as shown in figure 2.3. The model identification through machine learning techniques has a fundamental impact in the context of production systems enabling the use of the models for improving optimization tasks, energy efficiency, effectiveness of production plans while supporting predictive maintenance.



(a) Supervised learning (b) Unsupervised lerning

Figure 2.3: General machine learning formal framework

Types of learning	Paradigm	Description	Task	Algorithm
Supervised Learning	learn-from-observation	The aim is to learn a mapping from the input to an output (labeled example data set) and use this to create a general model for the problem	Classification and Regression	Naïve Bayes, Support Vector Machine, Decision tree, Rule Induction, Neural Network, Nearest Neighbour
Unsupervised Learning	learn-from-observation	The aim is to discover the regularities in the input (unlabeled example data set) finding statistical values to describe the data	Density estimation	Neural Network, Self-Organizing Map (SOM), Adaptive Resonance Theory
Semi-supervised Learning	learn-from-observation	This technique falls between the supervised and unsupervised learning by combining labeled and unlabeled example data set.	Classification and Regression	S3VM, Multiview Algorithms, Generative Models, Graph-Based Algorithms
Reinforcement Learning	Exploration and Exploitation	This technique relies on the best policy to apply. In some applications, the output of the system is a sequence of actions to reach a goal. Each action has always an impact on the environment. In such a case machine learning program should be able to calculate the sequence of actions with minor impact on the environment (with minor cost).	Goal-Oriented, Decision-Making	Q-Learning , SARSA, TD Learning
Transduction	Learn-from-observation	Related with the semi-supervised learning, transduction relies with the prediction of new outputs and/or conclusions based on a set of training inputs, training outputs and test inputs.	Classification, Regression	T Support Vector Machine
Learning to learn	Learn-from-observation	Similar to traditional inductive machine learning techniques that induce general functions from example set, Learning to learn approach tries to learn how to generalize the main problem.	Classification and Regression	Neural Networks, Genetic Algorithms
Explanation-based learning	Explain general from particular	This techniques consists on learning a general problem-solving technique by observing human solutions to a specific problem.	Classification, Regression	EGGS Algorithm

Table 2.1: Learning Algorithms Taxonomy

The *semi-supervised learning* approach falls between the *supervised* and *unsupervised learning* attempting to exploit the benefits of both approaches making use of both labeled and unlabeled data, i.e. using a training set with and/or without input-output pairs. Therefore the *semi-supervised learning*, addresses some drawbacks of *supervised* and *unsupervised learning* such as the problem of collecting labeled data that sometimes is difficulty, expensive and time consuming while improving, thanks to the labeled data, the performance of typical *unsupervised* classifiers/predictors [Zhu, 2005].

In *reinforcement learning* the idea of interaction between machine and environment is exploited. Inspired by the action-reaction law, this technique relies on the best policy to apply according to the given situation. As focused by Dayan in his work [Dayan, 2002], *reinforcement learning* deals with how animals and artificial systems can learn to optimize their behavior in the face of rewards and punishments, i.e. how acting on the environment obtaining rewards and avoiding punishments. The system of rewarding and punishments will, then, guide the learning algorithm. This learning techniques is based on an "Exploration and Exploitation" paradigm since a learning agent will observe repeatedly the sate of environment, in particular all the relevant aspects to support the decision making process, and taking into account both its own experience (Exploitation) and observations (Exploration) will chose actions so as to maximize rewards and minimize punishments.

The *transduction* or *transductive inference* was introduced by Vladimir Vapnik in [Vapnik, 1996] performing a new reasoning approach, distinct from the *induction-deduction*.

In machine learning domain *transduction* means reasoning from observed, specific training cases to specific test cases; it represents an inference mechanism from particular to particular, while *induction-deduction* means reasoning from observed training cases to general rules, which are then applied to the test cases. The big drawback of *transduction* is that no predictive model is built but only a mapping representative of the data is created. The relations between *transductive* and *induction-deduction* inference mechanisms are depicted in the figure 2.4.

Another research direction within the mainstream of the machine learning field is represented by the *learning to learn*. Inspired by human learning, the *learning to learn* approach is not only interested in the learning task but investigates also the possibility to learn how to generalize introducing a learning bias, which is chosen based on experiences, just like humans that are able to generalize correctly from extremely few examples [Thrun, 1996].

Finally, the *explanation-based learning* aimed at finding plausible and generalized explanations for why a particular example is an instance of a concept [Minton et al., 1989]. These generalized explanations were then converted into recognition rules that would be used to discover similar examples and/or to reason by analogy. The explanations are typically built in a deductive manner, i.e. directly from the examples without creating a general model for the problem.

Several studies have focused on using this techniques in the most disparate contexts such as finance (forecasting) [Ahmed et al., 2010], Internet traffic monitoring [Yuan et al., 2010], mobile robots [Mitchell et al., 1993], manufacturing (specially in production planning and enterprise management)[Tsatsoulis and Kashyap, 1993, Monostori, 2003, Carbonneau et al., 2008], and science (speech and vision recognition) [Dietterich and Flann, 1997, Kubat et al., 1998, Bishop, 2006, Hsieh, 2009]. However application of such solutions in industrial practice are not well explored.

In this context, Self-Learning solution intends to provide a generic architecture enabling application of machine learning techniques, for both control and management, in industry for improving manufacturing processes operation along time.

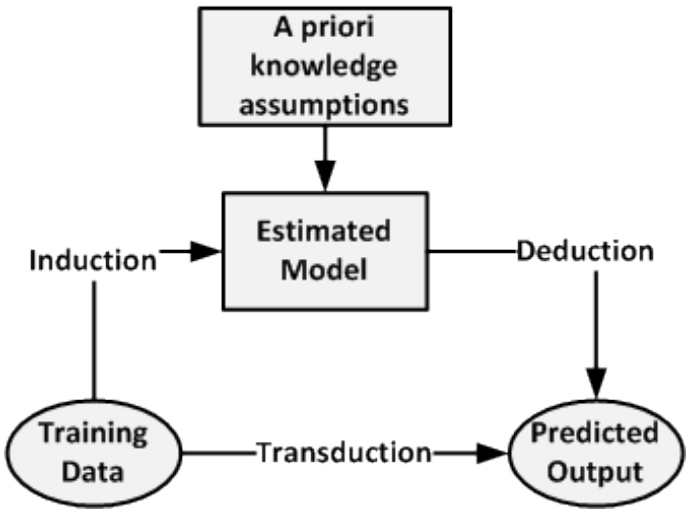


Figure 2.4: Main inference mechanisms parallel: *induction-deduction* and *transduction* [Cherkassky and Mulier, 2007]

2.4.2.2 Data Mining in Manufacturing

In recent decades, manufacturing enterprises have invested heavily in both Enterprise Resource Planning (ERP) systems and in automation for improving their production processes. However to reap the rewards of all these investments integration between the business world and the control and monitoring world is necessary. In this context, numerous standards (ANSI/ISA-95 [1], ISO TC184 SC5 WG1, CIMOSA [2], GERAM [Force, 1998]) from various industry and government groups have been developed to allow integration between enterprise and control systems in order to reduce the risks, costs, and errors that go hand in hand with implementing interfaces between such systems [Scholten, 2007]. All these standards are not automated systems itself, but define a method, an abstraction, a way of thinking for enabling enterprise software applications interoperability, easy integration and simplify information exchange between them while

[1] see http://www.isa-95.com/
[2] see http://www.cimosa.de/

improving the whole enterprise visibility. The common point between such standards is represented by a general definition of how to manage all the huge amount of information generated in manufacturing enterprise. In fact, within manufacturing companies, different kinds of information are produced at different manufacturing enterprise layers and typically stored in databases. Modern manufacturing enterprises are characterized by a deep dissemination of databases to guarantee that all the produced information can be accessed. As stated in [Han and Kamber, 2006], the use of databases is well established in engineering. However, the stored information is sterile if not supported by tools and techniques allowing to extract knowledge from raw data. In this scenario, the advancements in information technology (IT), data acquisition systems, and storage technology as well as the development in machine learning tools, algorithms and methodologies have solicited the research community to move toward discovering knowledge from databases (KDD), since databases in manufacturing companies offer enormous potential for transforming data into useful knowledge [Harding et al., 2006]. Moreover, the extracted knowledge can be used for classification tasks, modeling tasks, and to make prediction about future evolution of the analyzed variables. The idea of finding patterns from apparently unstructured data in manufacturing is not new, as a matter of fact the application of data mining techniques to manufacturing scenario began in the 1990s [Irani et al., 1993, Seabra Lopes and Camarinha-Matos, 1995, Monostori et al., 1996]. Data mining can be applied at different areas in manufacturing such as quality control, fault detection, scheduling and decision support. However, there are areas where data mining techniques are not exhaustively explored such as manufacturing planning and shop floor monitoring and control. To enhance the usage of data mining in industrial context the development of a standard methodology is necessary for allowing reliability and repeatability of data mining processes. The Cross Industry Standard Process for Data Mining (CRISP-DM) project [Wirth and Hipp, 2000] is an effort in this direction, proposing a comprehensive process model for carrying out data mining projects. The process model is shown in figure 2.5 and is independent of both industry/industrial application and the particular technology used.

The life cycle of a data mining project is broken in six phases, namely: *Business understanding*, *Data understanding*, *Data preparation*, *Modelling*, *Evaluation* and *Deployment*. The *Business understanding* represents the initial phase of the data mining project and is intended to gather necessary information for understanding objectives and requirements from a business perspective. All this knowledge is used to design an action plan to achieve the defined objectives and requirements. The second phase is the *Data understanding* that is intended to get familiar with the data, i.e. analyze the data to detect interesting subset to discover hidden information. The first two phases are strictly related since to create an action plan a general understanding of the data mining problem from the data point of view is fundamental. The *Data preparation* phase is intended to perform all the activities to construct the final dataset representing the input of the data mining

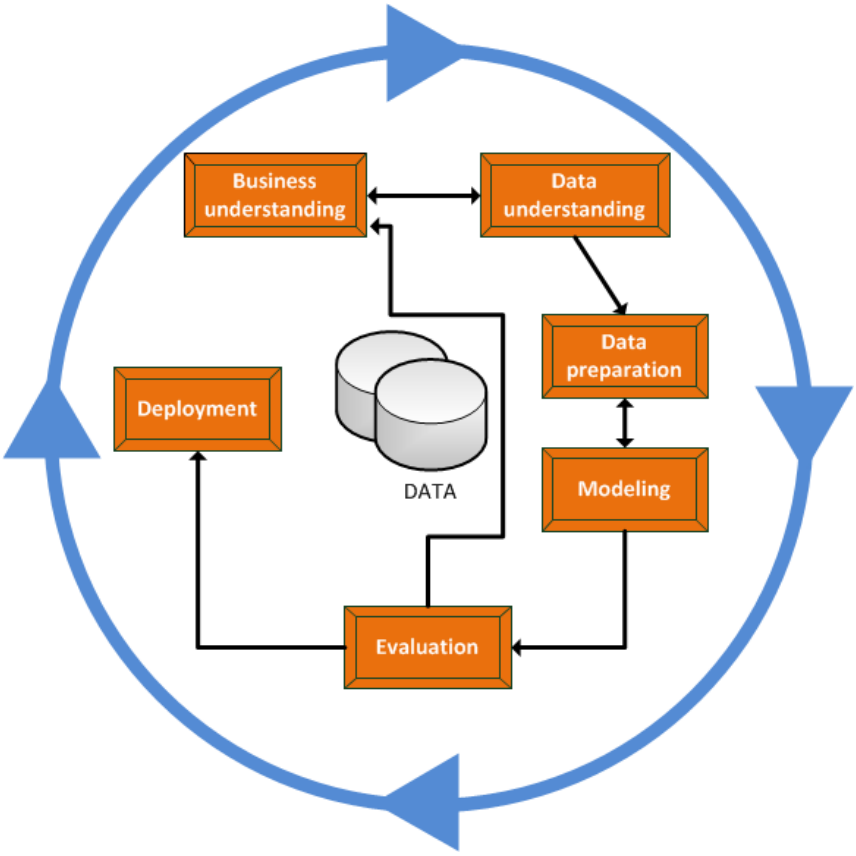


Figure 2.5: Generic CRISP-DM process model [Wirth and Hipp, 2000]

problem. The *Modelling* phase is intended to build a model of the problem from the available data, this model is then used to predict/classify new situations. Likewise the first two phases, the *Data preparation* and the *Modelling* phases are related since the model capabilities rely on the quality of the data selected during the *Data preparation* phase. The *Evaluation* phase is intended to evaluate the model and eventually refine it in order to be sure that the data mining application is able to achieve the defined objectives and requirements. Finally, the *Deployment* phase is intended to organize the gained knowledge and show it in an appropriate manner to the user of the system. Although, the details of each step of CRISP-DM methodology make it easy to use and fast to implement further research is needed to develop generic guidelines to be applied to different problems and kind of data while improving integration of the methodology in control and monitoring systems design for manufacturing processes.

Despite the existence of standards processes for data mining, most of the data mining applications for industry are stand-alone, one-of-a-kind and not fully integrated applications inside manufacturing-based enterprise reference architectures, frameworks, middleware and standards control and monitoring solutions. Several data mining applications have been developed in different areas of manufacturing. Taking into account the

main focus of this dissertation, areas such as decision support in job scheduling/dispatching, energy/maintenance activities, quality product improvement and optimization of manufacturing process monitoring and control at shop floor level are considered. In each one of these areas, several research projects aiming to provide data mining valid solution for manufacturing have been analyzed for carrying out current trends of industry and research community toward applications of data mining. In [Huyet, 2006] an optimization methodology for job scheduling/dispatching in manufacturing production systems based on an evolutionary and data mining approach has been proposed. Li and Olafsson in [Li and Olafsson, 2005] introduce a novel methodology for discovering unknowing scheduling/dispatching rules using a data mining approach applied directly to the generated production data. In [Batanov et al., 1993], a prototype called EXPERT-MM has been developed which works on historical data gathered during production activities and provides suggestions for an appropriate preventive maintenance scheduling. In [Romanowski and Nagi, 2001], Romanowski and Nagi applied data mining approach in a maintenance domain to detect the subsystems responsible for low equipment availability. Once recognized, a recommendation for preventive maintenance interval is made. Finally, at shop-floor level a huge amount of data is generated during production activities, the usage of data mining approaches at this level allows the extraction of new knowledge about manufacturing process ensuring a better characterization of the general process and improving the final process outcome. Da Cunha in his work [Da Cunha et al., 2006] applies data mining techniques to the production data to determine the sequence of assemblies activities that minimize the risk of producing faulty products. In [Chen, 2003], a cell-formation approach based on association rule induction is developed to find effective configurations for cellular manufacturing systems. Chien in his paper [Chien et al., 2007] develops a framework aiming to investigate the huge amount of semiconductor manufacturing data and infer possible causes of faults and/or manufacturing process parameters variations in order to refine monitoring and diagnostic information. Gardner and Bieker in [Gardner and Bieker, 2000] use a combination of self-organizing neural network and rule induction to identify the critical poor yield factors and quality problems analyzing collected wafer manufacturing data. The problem is a typical intermittent and non-linear problem making the detection of yield and poor quality problem difficult. Applying data mining technology to this context improves the whole system capability to detect these problems while enabling the adaptation of the manufacturing process parameters. In [Charaniya et al., 2010] data mining techniques has been applied to the manufacturing process of bioproducts for comprehending the complex characteristics of bioprocesses and enhancing production robustness through the analysis of a vast amount of data gthered during production activities. Finally, Srinivas and Shahbaz in [Srinivas and Shahbaz, 2004] have developed an agent-based framework for highly flexible shop floor control system architecture enhanced by the application of data mining approaches for knowledge discovering from databases to make fundamental decisions aimed at optimizing the enterprise objectives.

Knowledge discovery and data mining techniques applied to manufacturing systems enable the extraction of useful information and knowledge from manufacturing databases. The huge amount of data stored in databases during production operations represents a great source of potentially new information that need to be explored to enhance fundamental manufacturing activities such maintenance, optimization of process parameters as well as scheduling/dispatching of resources at all the manufacturing enterprise levels, i.e. from business to control level. The literature review shows several applications in this direction confirming from one side an exponential growth of data mining application but from the other side the lack of a generic holistic approach and design methodology as well as a reference architecture for constructing easy-integrable data mining applications.

Finally, the growing interest aroused by data mining and knowledge discovery techniques applied to manufacturing systems has been also demonstrated by several research projects funded by the European community, underlining the importance that these techniques have for the development of the factory of the future. Some ongoing European R&D projects that are grounded on the deep use of data mining are plotted on the time bar in figure 2.6 considering the starting date.

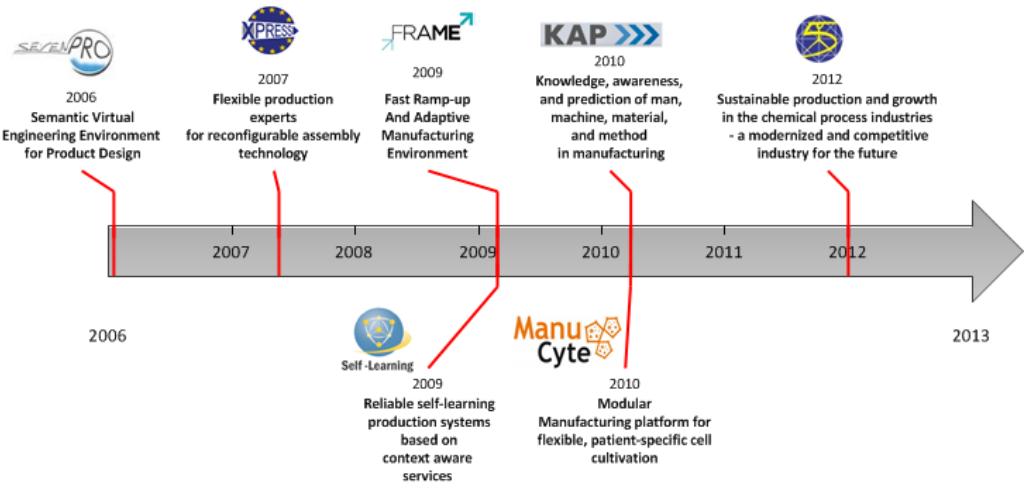


Figure 2.6: Research activity in manufacturing applications of data mining over time

2.4.3 Industrial standards for manufacturing production process automation

Today manufacturing systems are characterized by complexity since they are constituted by an enormous number of different devices from different suppliers (sensors, actuators, controllers, robots and so on) that are all involved in the production of an increasingly complex products. In this scenario cooperation is needed within and between them in order to guarantee the correct execution of all the steps of production cycle that leads to the creation of goods. As stated in [Barata, 2005], cooperation can only be achieved by fast, reliable, and updated information exchange since without any kind of communication and/or information exchange does not make sense talking about cooperation and,

thus, interoperability between devices.

The increasing distribution of components devices into automation systems and the exploding amount of information require more and more increasingly complex interfaces for operation and monitoring [Schwarz, 2005]. In order to reduce the complexity and understand how interfaces interact, i.e. how different components can interact with each other, it is important to have standards describing in a well-defined mode how interfaces work so that the various systems fit together. Standardization is probably the immediate answer to any problem that involves interoperability between systems and devices. Control software represents the main element in today's industrial automation system for providing correct and safe operation of the whole automation process. As referred in [Hajarnavis and Young, 2008], all manufacturers had standardized the development of their systems across plant worldwide, encompassing electrical systems and diagnostics as well as Programmable Logic Controller (PLC) types and software structure. In order to achieve this, each manufacturing company has developed a set of company-specific documentations concerning how to develop and/or implement control and supervision software systems. Standardization of control and supervision software systems were focused on the following main points, as pointed by Hajarnavis in [Hajarnavis and Young, 2008] and by Zoitl in [Zoitl and Vyatkin, 2009]:

- Introduce a common approach for developing and implementing control systems covering both hardware and software aspects with the purpose of reducing commissioning time;
- Reduce the development time of industrial and automation control applications by reusing developed control software elements across different automation projects and also across control devices of different vendors;
- Improve machines/system diagnostic;
- Support the process of change, software understanding and maintenance activities;
- Reduce significantly the costs for personnel training; and
- Improve the software quality and reliability;

Therefore, standardization issues are related with defining a common way of handling inputs, outputs, data types and control programs while providing a well-defined set of practices to be used during the process of software building and structuring.

This standardization phenomenon is strongly present in the today industry and current development trends in industrial control and monitoring software call for increasing standardization of software. However, since the software structure is defined by the end-user rather than individual programmers then the flexibility of the programmer is limited, making harder the process of introducing new technologies and/or approaches.

In this scenario the Self-Learning project is intended to resolve this problem by developing a software architecture and solution completely decoupled from the existing control and monitoring software systems and in such a way independent from the particular manufacturer standard. Therefore, the SLPS solution will be capable to work in parallel with the main manufacturing process control without interfering its control activities.

The subsections 2.4.3.1 and 2.4.3.2 will present two different standard monitoring and control system approaches for the automotive domain to frame the idea that each manufacturer has its own standard and way to operate even if the manufacturing processes are basically the same.

2.4.3.1 Ford Motor Company standard

The main OEM role is to integrate systems from different suppliers in order to create a solution capable to produce complex products. This aspect underlines how important is to have well-defined interfaces so that the various systems fit together. As stated in section 2.4.3 current development trends in industrial monitoring and control solutions call for increasing standardization of software. As an example Ford Motor Company develops a set of software specifications so called Diagnostic Control Program (*DCP*) to ensure different vendor's software to implement well-defined monitor and control solutions.

The main challenges addressed by the *DCP* standard are the following:

- definition of a standard approach to the control of sequential treatment-processes, integrating dynamic diagnostic information about the machine status, not only in the error case;
- software project has to be well-structured, organized and documented in order to be easy to understand; and
- the software program should allow unequivocal fault finding and easy machine restart after a fault condition.

2.4.3.1.1 The *DCP* standard Approach

The *DCP* standard approach consists on modelling the behavior of each machine/station in order to decompose it into functional sequences that will in turn constituted by execution steps. Each execution step is then monitored, i.e. the correct sequence of input/output signals is monitored during its execution. Therefore, the sequence-step logic models a finite state machine where the transition from one state to the next depends by the occurrence of certain conditions (see figure 2.7). The using of a diagnostic fault table stored into PLC memory of each manufacturing station, to keep the information regarding the actual state of a machine and/or station during the execution, is an integral part of the control program. The table is organized into several zones each one containing bits that are controlled by signals with different features, i.e. signals that are monitored

only in required periods (e.g. limit switch) and/or continuously (e.g. Emergency Stop). An empty table, therefore, indicates that there are no faults present and that the machine/station may continue further in its operation. The control program will use the diagnostic information stored into the diagnostic fault table to control the machine/station sequence.

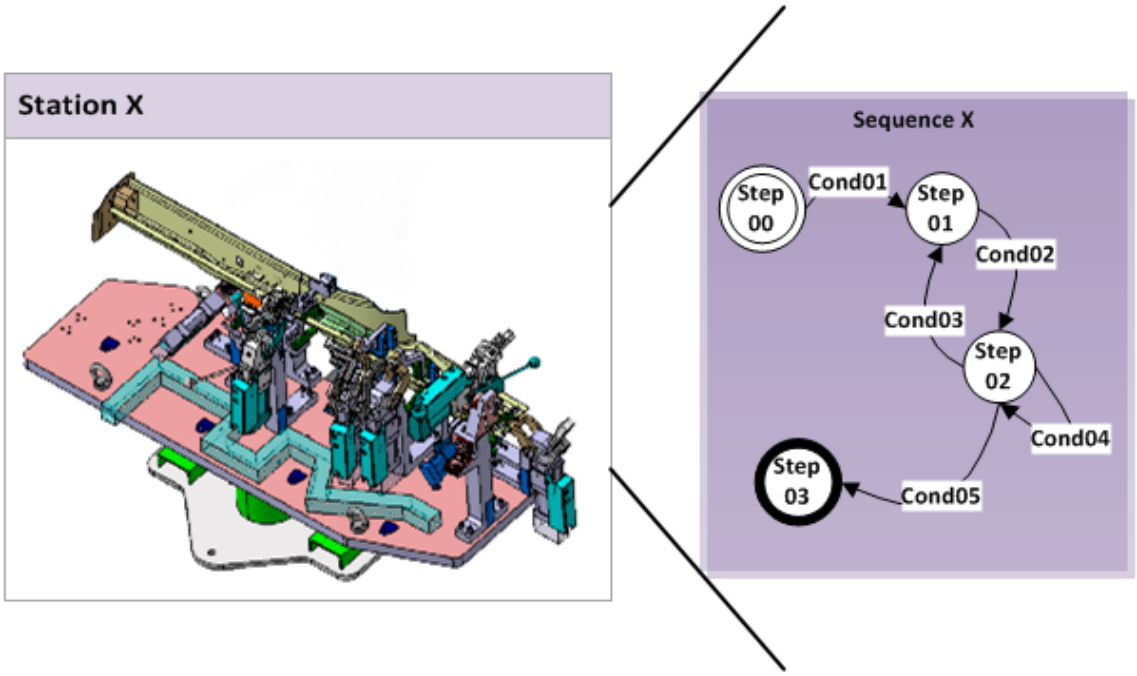


Figure 2.7: *DCP* standard approach for control

2.4.3.2 Volkswagen AG standard

The Volkswagen AG has developed a set of standard directives and instructions grouped into one document (*VASS* standard project configuration directive) for guiding different equipment suppliers during the implementation of its monitoring and control solutions, both for hardware and software configuration.

The main goal of the *VASS* standard is to provide a general and unique way for programming and configuring monitoring and control solutions for Volkswagen group manufacturing processes, ensuring:

- High flexibility and quality of the entire system;
- Minimization of investments in terms of money and time for reengineering tasks during the system lifecycle;
- Well-structured, organized and documented solutions to make easy for the maintenance personnel to work with them;
- Modularization of control system into manageable and replaceable units; and

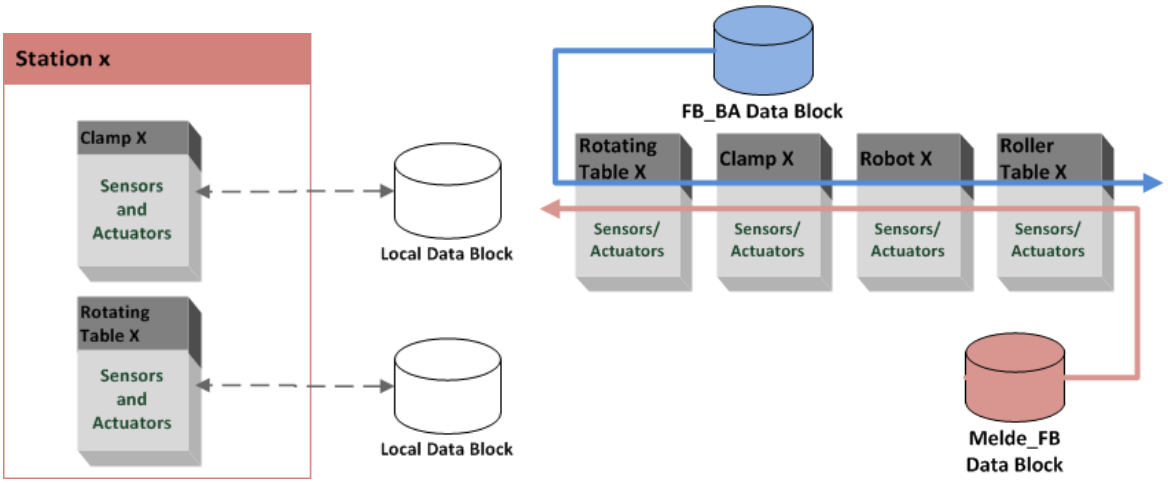
- Simple standard interfaces between all the general architecture modules.

2.4.3.2.1 The *VASS* standard Approach

The *VASS* standard provides a modularized approach, based on function blocks, for building monitoring and control solutions while enhancing flexibility and modularization. All the function blocks encapsulate certain functionality(s) representing hardware devices in a manufacturing line. Since a machine/station is constituted by several interconnected hardware devices then machine/station function blocks will be in turn constituted by several elementary function blocks each one representing the resources or devices into the machine/station.

Since the elementary function blocks are self-contained software capable to control hardware devices all the monitoring and control variables are, then, stored into data blocks associated to each one of the distinct function blocks (see figure 2.8(a)).

To allow the communication of global system information (e.g. operating mode availability, fault messages, etc.) that are common to all machines/stations inside the particular manufacturing line, two standardized function blocks with associated data structures gathering all this information are used as backbone (see figure 2.8(b)). The *FB_BA* data block stores relevant global information such as 24V power supply availability, air-compressed power supply, operating modes general status, etc.. The *Melde_FB* data block stores relevant information for faults management.



(a) Local Data Block associated to each function block

(b) Communication of global system information

Figure 2.8: *VASS* standard approach for control

Finally, the *VASS* standard has the function block as fundamental core component and enables integration and communication between them using a set of shared data blocks.

2.4.3.3 Consideration on Practices for PLC Software Development in Industry

The sections 2.4.3.1 and 2.4.3.2 aim to explain that similar manufacturing processes can be monitored and controlled using a totally different approach. However some similarity can be found into the two presented approaches starting from the device used to control the manufacturing process. The PLCs are established as the device of choice for the implementation of control algorithms in many manufacturing companies [Hajarnavis and Young, 2008]. The development of PLC programs relies on the usage of the five languages of the IEC 61131-3 [Lewis, 1998] standard. However as exposed in [Lucas, 2003] and recently confirmed by a poll realized by the Control Engineering U.S. and Control Engineering Poland magazine [Pietrusewicz and Urbanski, 2011] 96% of industrial application are developed using the ladder logic diagrams (the DCP and VASS approaches fall in this group). The use of the ladder logic diagrams (LLDs) for developing monitoring and control solutions implies several drawbacks. As a matter of fact, the low level of representation, semantic and poor data structure typical of the LDDs lead to low monitoring capabilities and reduced opportunity of optimization during production processes lifecycle, since the exchange of simple binary information does not allow to create a comprehensive overview of the entire system and its related processes. The poor monitoring capabilities and the lack of truly optimization activities in terms of energy efficiency and preventive maintenance do not depend from the particular approach used to program PLCs (even if the use of one approach in repect to another could improve the monitoring a little) but is directly related to the LLDs limitations. In this domain the adoption of software architectures based on new methodologies and technologies such as SOA, Agents, Machine Learning, etc. in addition to the traditional monitoring and control solutions may leads to enourmous benefits regarding efficiency, optimazion, adaptability and fault tolerance, improving at the same time the industrial community acceptance.

2.4.4 Service Oriented Architecture (SOA)

2.4.4.1 Definitions and basic concepts

SOA [Erl, 2006, Josuttis, 2007, Papazoglou and van den Heuvel, 2007] is a relatively new term representing an emerging approach that addresses the requirements of loosely coupled, standard-based, and protocol-independent distributed computing.

The SOA are being promoted as the next evolutionary step to help organization to meet the more complex challenges imposed by globalization and markets fragmentation. It establishes an architectural model that aims to enhance the efficiency, agility, and productivity of an enterprise by positioning services as the primary means through which solution logic is represented in support of the realization of the strategic goals associated

with service-oriented computing [Erl, 2006]. It provides a framework or platform for realizing rapid system development, easily modified systems, while enhancing system integration capabilities and overall system quality. Using the definition given by Komoda in his work [Komoda, 2006], a Service Oriented Architecture is a design framework for construction of systems by combination of *services* and using deeply ICT infrastructure as communication backbone. SOA paradigm is inspired by the provider/client interaction typical of today's human society, where the provider and the client coordinate their works to both create and capture value. Hence, SOA envisions the development and implementation of a platform consisting of independent *services* representing well-defined and self-contained modules providing standard operations that can be invoked by internal or external components (consumer of the *services*) using standard interfaces. The *services* can be combined and recombined into different solutions and scenarios, since they do not depend from the state and/or the context of the other services.

Therefore, the advent of SOA paradigm and its core building blocks (*services*) promises to continue to radically change the way the different system components interact with each other, enabling true *interoperability* and system *scalability*. The former is related to service execution on hetero-environment thanks to standard based interfaces that is independent of implementation technology, while the latter is related to the capability to add and remove services without affecting the entire infrastructure. The main characteristics of the *services* that compose a SOA have been brought to light by Channabasavaiah in his work [Channabasavaiah et al., 2003] in the effort to create a better understanding about SOA and its feasibility, and can summarized as follows:

- All the *services* that compose a SOA are autonomous, i.e. their operation are perceived as opaque by any external component that is interested to use it meaning that external components neither know how services perform their own internal function but they are interested to receive the expected result. Therefore, the implementation and execution of services are, then, hidden behind the service interface;
- The interfaces of services must be always invocable, emphasizing the idea that should be irrelevant if services are local or remote, the interconnect schema and/or protocol to effect the invocation and the which infrastructure components are required to establish the connection.

The interface of *services* is the key element of a SOA showing how to invoke the desired operation provided by local or remote components inside the architecture. It defines the necessary parameters for the calculation of the result as well as the type of the result, meaning that the nature of the *service* is explained in its interface without any reference to the technology used to implement it. However, the feasibility of SOA directly depends on technological issues. In this scenario many technologies have been used to allow integration rather than communication between distributed component systems within an enterprise. Some of the main technologies used to support interoperable machine-to-machine interaction are:

- **Socket** - A socket represents a communication channel between two end-points allowing an application to connect to the network and exchange data with another application connected to the same network. Although several good working implementations of sockets exist, this approach has two major drawbacks, namely the need to know exactly the data format and the detailed location of the ends-point.

- **CORBA** [Corba, 1995] - The Common Object Request Broker Architecture is a middleware that provides an infrastructure for distributed application integration. The architecture still suffers some limitations such as high complexity, incompatibility between different vendor distribution, and above all significant challenge for Web-enabling these applications as well as for ensuring a safe communication as pointed in [Murray and Golluscio, 2002].

- **COM/DCOM** [Frank and Redmond, 1997] - DCOM is the acronym for Distributed Component Object Model and was introduced by *Microsoft* in 1996. Similarly to CORBA, DCOM is a middleware developed for enabling integration between networked applications and represented its major competitor. However, several limitations such as severe security problems and the need to use Windows based systems have reduced its practical use.

The existence of *Web Services* technology has enabled and stimulated the implementation and development of SOAs, i.e. the transposition of human society provider/client paradigm to the software applications world. As stated in [Natis, 2003], although *Web Services* do not necessarily translate to SOA, and not all SOA is based on Web services, the relationship between the two technology directions is important and they are mutually influential: *Web Services* momentum will bring SOA to mainstream users, and the best-practice architecture of SOA will help make *Web Services* initiatives successful. However an important distinction between *Web Services* and *Services* has to be made, since the two terms can be confusing. Reusing the words of Barry in his book [Barry, 2003], the term *Web Services* refers to a collection of technologies such as eXtensible Markup Language (*XML*) [Bray et al., 1997], Simple Object Access Protocol (*SOAP*) [Box et al., 1999], Web Services Definition Language *WSDL* [Christensen et al., 2001] and Universal Description, Discover and Integration (*UDDI*) [Bellwood et al., 2002] (see figure 2.9). *Web Services* provide a standard means of interoperating between different software applications, running on a variety of platform and/or framework and, thus, provide a set of functionalities independently from the particular hardware and technology used.

Whereas, *Services* are what you connect together using *Web Services*, meaning that *services* represent the endpoint of a connection. According to the figure 2.9, the steps involved in providing and consuming a service when *Web Services* technology is used are the following:

1. A service provider has to describe its service or list of services using a *WSDL*. The definitions are, then, published or discovered any where on the Web.

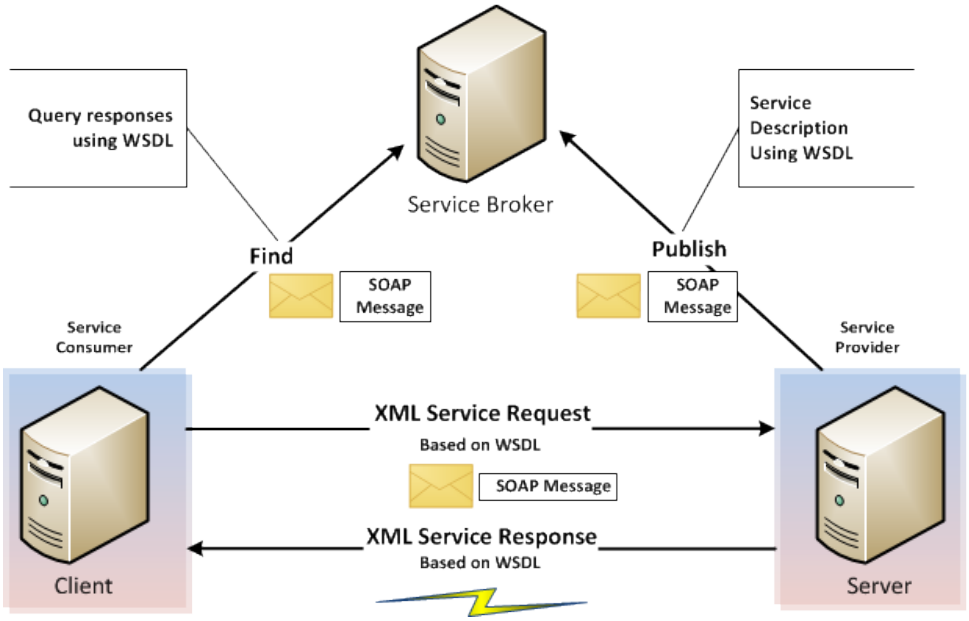


Figure 2.9: *Web Services* basic components

2. A service client can query the directory or network in order to locate the best suitable service according to its purpose.

3. Part of the *WSDL* provided by the service provider is, then, passed to service client. This contains all the necessary information on how to invoke the selected service.

4. The service consumer will send a request to the service provider based on *WSDL* information.

5. The service provider (when requested) will execute the operation and, for the case of request-reply, provide the final result to the service consumer.

Finally, despite the particular technology used to implement and develop SOA solutions, the SOA is a design philosophy embodying the next wave of distributed system development. The visionary promise of SOA consists in the possibility to easily assemble application components into a network of loosely coupled services stressing interoperability and location transparency. SOA raises enterprises to a new level of dynamism while empowering their flexibility and agility. As a result, SOA is becoming the *de facto* standard for designing and developing highly reliable distributed systems.

2.4.4.2 SOA in Manufacturing

As explained in section 2.1, future manufacturing enterprises will act and compete in a new challenging environment characterized by frequently changing in markets demands, reduced time-to-market, increasing consumer demand for highly quality and customized products at low cost. However, as exposed in [Jammes et al., 2005], production costs and

quality of products remain vital concerns meaning that the only freedom degree is represented by reducing time-to-market. This demand obliges manufacturing enterprises to include the password "change" in their own architecture and infrastructure to improve competitiveness in market sharing. Change in any organization can be applied to both organizational and technical level. Assuming the enterprise software infrastructure organization depicted in [Mathes et al., 2009], three vertical layers are adopted, namely: *business layer*, *intermediate layer* and *manufacturing layer* (see Figure 2.10).

The *business layer* of an enterprise contains software functionality related with accounting, human resources, administration, marketing and monitor markets demands.

The *intermediate layer* is responsible for receiving production orders coming from the *business layer*, processing it and generate command for the *manufacturing layer*. Furthermore, it gathers data information about the status of the *manufacturing layer*, i.e. status of the manufacturing assembly line and current production behaviour/context.

The *manufacturing layer* handles the manufacturing production process. Modern shopfloors are characterized by a high degree of diversity in device functionality, form factor, communication protocols, input/output features, as well as the presence of many heterogeneous and often proprietary software and hardware components as focused in [Cannata et al., 2008].

The ICT infrastructure of a manufacturing enterprise is highly heterogeneous and distributed, ranging from the *business layer* with standardized ERP solutions, to the *manufacturing layer* with standardized vendor-specific automation solutions passing through the *intermediate layer* where the Manufacturing Execution System (MES) used depends on the equipment installed in the *manufacturing layer*. The vertical adoption of SOA and *Web Services* on these three different layers facilitate the data integration and provide a degree of agility not afforded with the previous attempts using CORBA or COM/DCOM solutions. This agility guarantees an easier and faster responsiveness and reactiveness of the whole enterprise than before, allowing to face better the challenges imposed by a competitive environment dominated by change and uncertainty as defined in [Goldman et al., 1995].

However as stated in [Cândido et al., 2009], the agile performance of a manufacturing enterprise is limited by its least agile building block, meaning that to be agile all enterprise ICT levels, from business to device and/or manufacturing level need to be agile to interact in a seamless and synchronized manner. Several paradigms has been thought to improve flexibility and agility inside manufacturing enterprises. As exposed in [Ribeiro et al., 2008], Multiagent Systems (MAS) and SOA are the most promising approaches for developing these paradigms even if they have their own peculiarities, i.e. strengths and weaknesses making them most suitable for certain applications. Nevertheless in the context of Self-Learning project, the nature of the problem and the will to have a solution to enable tight integration between the *business*, *intermediate* and *manufacturing* layers of a manufacturing enterprise render the SOA paradigm the best choice.

The application of SOA and *Web Services* in the context of *manufacturing layer* is still scarce, since a set of persisting technical challenges exists as pointed in [Ribeiro et al., 2011].

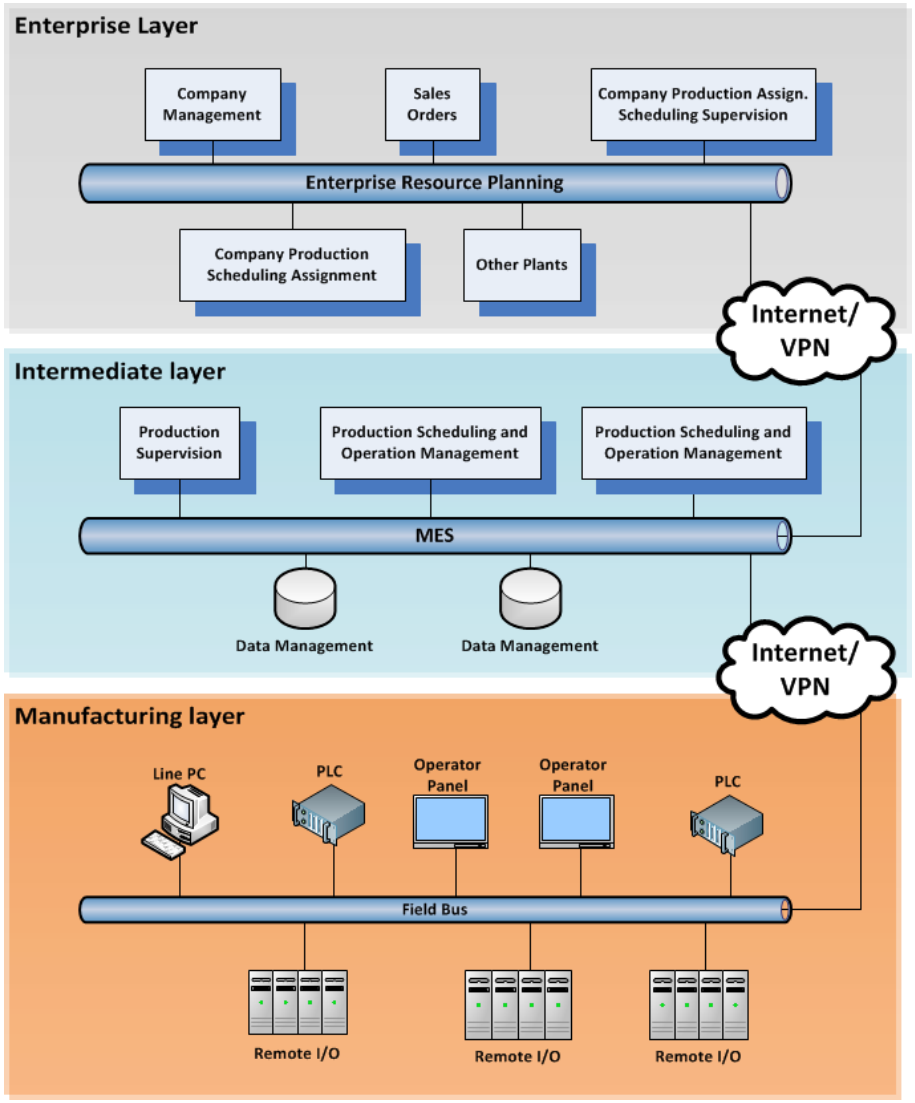


Figure 2.10: Typical Manufacturing enterprise software organization

The lack of a unique reference standard architecture, which includes all the set of practices and procedures necessary for the deployment of the system, affects negatively the dissemination of the SOA paradigm in the monitoring and control world. Furthermore, devices in *manufacturing layer* should be designed and developed taking into account the new requirements imposed by SOA paradigm with a particular attention to the interface design phase. Moreover, the typical performance constraints of manufacturing production lines exclude the use of managed languages in hard real-time control due to a trade-off between the abstraction of the language and the execution time. Finally, social and educational aspects have a fundamental impact on the dissemination of SOA paradigm. Workers with a strong background in logic based control can be hostile in the introduction of the new paradigm and unable to handle a system partly designed by computer engineers and whose functioning premises are far from conventional shop floor knowledge. All these challenges has been tackled by several authors [Jammes et al., 2005,

Colombo et al., 2005, Lastra and Delamer, 2006, Cachapa et al., 2007, Cândido et al., 2011] and research initiatives SIRENA [3], ITEA SODA [4], IST SOCRADES [5], AESOP [6].

However only a few part of the results were really applied and used in real industrial scenarios during the production activities implying that more work is needed. Moreover, developments trends in data mining technology and applications analyzed and underlined in section 2.4.2.2, demonstrate an increasing awareness of industrial community in respect of data mining application for problem solving in manufacturing. However also in this case, most of the reviewed applications and solutions are one-of-a-kind and, above all, highly specialized in solving a particular problem. Fundamental features such as genericity, abstraction, portability and easy communication/integration are not considered as important ones implying that the developed solutions cannot be applied at all the levels of a manufacturing enterprise and at different industrial application scenarios. The Self-Learning project tries to fill these gaps and to face these challenges by providing a new methodology for the design and development of monitoring and control solutions for manufacturing companies relying on context awareness and data mining techniques. The SLPS approach is intended to be detached from the application context allowing its usage in different kind of production systems and environment with small programming effort. Finally, the SLPS solutions are grounded on an efficient distributed service oriented communication infrastructure enabling easy integration with existing knowledge based systems and monitoring and control architectures while improving their capabilities and pushing them to a new level where awareness, responsiveness, adaptability, control and maintenance efficiency and learning by experience are the key aspects.

[3] see http://www.sirena-itea.org/
[4] see http://www.soda-itea.org/
[5] see http://www.socrades.eu/
[6] see http://www.imc-aesop.eu/

3

Self-Learning Production System Reference Architecture

This Chapter aims to depict the overall reference architecture for Self-Learning Production Systems focusing the attention on the two main kernel components, namely the *Extractor* and the *Adapter*; a description of their features, functionalities and responsibilities as well as of the interactions between them is provided in order to frame their role inside the overall system. However, throughout this chapter much more attention is given to the *Adapter* component, since it represents the core subject of this dissertation.

The reminder of this chapter is organized as follows: section 3.1 shows the requirements analysis that guided the design and development of the SLPS reference architecture; section 3.2 primarily presents the overall generic architecture on which the SLPS platform is built and secondly details the SLPS *Adapter* component together with all the aspects related with its features, functionalities and behavior within the SLPS architecture.

3.1 Requirements Analysis

The work described in this dissertation falls under the scope of SLPS project, which motivation relates with the strategic objective of strengthening European Union (EU) leadership in manufacturing production technologies in the global marketplace by developing innovative self-learning solutions to enable a tighter integration of control and maintenance of production systems as well as its integration within the other manufacturing enterprise layers, within the concept of extended enterprise. To face this need, a highly reliable and service-oriented platform has been designed to support self-adaptation of

the production system while enhancing integration. The design activity involved partners from academia, research and industry. The diverse effort resulted in the definition of a collection of general requirements that SLPS solution have to comprehend.

The general requirements are classified according to the categories proposed by ISO9126 [iso, 2001] (see table 3.1).

Requirements Taxonomy	
Category	**Description**
Functionality	identify all the functions that the system has to encompass for satisfying identified needs for the companies
Usability	identify all the system type of users as well as the degree of expertise and type of interaction in order to tune the effort for the use of the solution according with the rank of user
Security	define the limits and/or access rights to the solution and to the data stored inside all the repositories
Reliability	identify the capability of the solution to perform its own functions under stated conditions and for a specified period of time
Efficiency	identify requirements of time response, i.e. the capability of the solution to produce the expected outcome effectively with a minimum amount of time
Maintainability	identify requirements of easy modification to face new requirements and/or correct defects
Portability	identify requirements that enable the solution to be transferred from one environment to another that is different form the one for which it was originally designed

Table 3.1: Requirements Taxonomy according to ISO9126

This collection of requirements concerns to the general needs and/or conditions that SLPS solution has to encompass. However, since SLPS solution is driven by several industrial application scenarios from three industrial partners, the general collection of requirement will be particularized and refined by extracting further information from the different proposed application scenarios.

More information about SLPS solution general and application scenarios specific requirements can be found in [Self-Learning, 2010a].

3.1.1 Generic Requirements for SLPS

The proposed solution for SLPS has been designed to fulfill the following generic requirements organized in the taxonomy showed in section 3.1:

- **Functionality**: the SLPS solution should support the interaction with existing manufacturing process equipments in such a way as to make possible the collection of all the necessary data to frame the actual operational context of manufacturing process. Hence, the collected data should be used to adapt manufacturing process machine parameters. To enhance the adaptation of manufacturing process parameters,

i.e. to understand how the manufacturing parameters affect the production activities, the solution should be capable to consider both the actual and past operational contexts. The development of explicit representative model of the manufacturing process from empirical apparently unstructured data is, then, peremptory. Finally, a User Interface (UI) needs to be provided to allow exhibition of final output of the system as well as for validation/evaluation tasks performed by a human expert.

- **Usability**: the SLPS solution should be easy to understand with limited training effort and supported by a comprehensive help system for allowing any user (especially those without any computer expertise) to be able to use the system in a satisfactory manner. All the displayed information should take into account the expertise of the operator and the current application context. Finally, the solution should be fully integrated into the manufacturing enterprise software infrastructure, i.e. the solution should ensure access to all data and/or information handled.

- **Security**: The SLPS solution should limits the access rights to particular functionality while guarantying the privacy of the information and the protection of data during transmission.

- **Reliability**: The SLPS solution should enable continuous operation and, in case of failure, should resume its normal operating cycle.

- **Efficiency**: The SLPS solution should be a totally non-intrusive system, designed to improve legacy monitoring and control solutions without affecting their normal execution and to respond in an appropriate time frame.

- **Maintainability**: The SLPS solution should be designed and implemented following a modular and distributed approach enabling easy extension and minimum impact if changes in one module have to be implemented, as well as distribution of its parts on different machine/environment. Furthermore, failures during system operation should be detected with minimum effort.

- **Portability**: The SLPS solution should be generic enough to be easily portable between different environment while continuing to work as expected and, above all, ensuring minimal reprogramming needs.

The figure 3.1 resumes all the requirements that the SLPS solution has to attain.

The generic requirements for SLPS solution establish the guidelines for the design of the system and its core components from a generic point of view. However, other challenges arise when trying to integrate SLPS solution inside a real industrial manufacturing environment. In the context of Self-Learning project, three business cases from three industrial partners have been studied and considered to demonstrate the validity of the concept as well as of the solution itself. The preliminary studies about these business cases finally resulted into the definition of new specific-process requirements intending

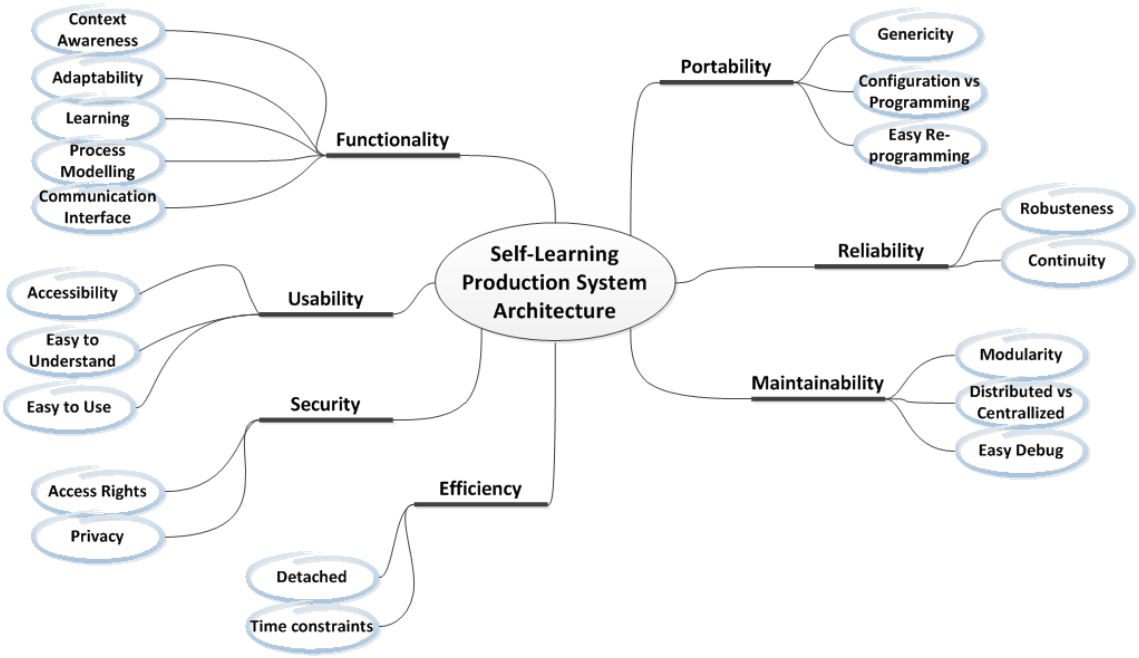


Figure 3.1: Imposed requirements for SLPS design

to refine all the generic components of the architecture for allowing the exploitation of the SLPS solution into the proposed application scenarios.

3.1.2 Business Cases specific Requirements

The purpose of this section is to identify process-specific requirements to allow the SLPS solution to meet the constraints derived by industrial business cases. Once the generic requirements for SLPS solution are discussed and a list of specifications is established, the next natural step is to identify the application context for SLPS solution and refine the previous list of specifications. Several topics including shop floor software and communication infrastructure, existing equipments, integration of new capabilities into existing monitoring and control architecture as well as data format and information flow from equipment to the higher levels of manufacturing enterprise are analyzed and discussed with the proper level of abstraction. The final result of this preliminary activity is a set of required add-on generic components and functionalities to enable the application of SLPS solution into real industrial context.

The Self-Learning project covers several application scenarios organized in three business cases each from a different industrial partner. The selected business cases are the following:

- **Business Case 1** from Bosch-Rexroth (BR): Self-Learning optimization of secondary processes in *CNC* machine tools.
- **Business Case 2** from Desma: Self-Learning intelligent monitoring and adaptation of machines parameters for shoe industry.

- **Business Case 3** from Fastems: Self-Learning scheduling and dispatching in Flexible Manufacturing Systems (FMS) for automotive industry.

The next subsections are intended to explore the three considered business cases with a view on main objectives and goals identifying, in such a way, a list of additional process-specific requirements for each one of them. Therefore, only a main description of the business cases (BCs) is given to frame the different context of application while allowing the reader to dip into the application scenarios. A more exhaustive explanation of the BCs and related application scenarios will be given in chapter 4.

3.1.2.1 Bosch Rexroth - Business Case 1: Optimization of secondary processes in Machine Tool

Today's manufacturing production lines, to deal with productivity and flexibility requirements, are typically constituted by several complex machines, each one with a set of tools and/or resources used according to the different production phases. As the complexity increases, new auxiliary processes are needed for ensuring the proper management of machines and related resources. Modern flexible machine, then, has several auxiliary and/or secondary processes related to its resources corresponding to a significant part of the energy consumers in manufacturing plants showing that there are yet some deficits in terms of sustainability. These processes are normally monitored and controlled by a local controller, allowing machine proper functioning along time. During the production process the local controller of the machine communicates with the manufacturing line controller sending machine status information and receiving commands, however this approach has the problem that is not able to provide comprehensive overview on the entire system and its related processes, since only simple binary information are exchanged. To improve machines energy efficiency is necessary to increase their efficiency factor, and above all is important to act on their utilization degree [Schmitt et al., 2011] reducing the energy consumption during unavoidable idle time periods. As a consequence, the application of new and more "intelligent" control and monitoring strategies is fundamental for improving machine preservation while decreasing the costs of production. BR has decided to follow in this path focusing on an holistic approach to tackle the optimization of all relevant production activities.

Hence, the presented scenario relates with the **optimization of secondary processes on CNC machines, such as maintenance and energy efficiency activities, during the machine tools lifecycle, integrating SL solutions to the existing service platform** as introduced in [Bittencourt et al., 2011]. Therefore, the goal in this business case is to improve manufacturing lines sustainability by using context aware and self-adapting solutions provided by the SLPS approach.

Considering the main goal of the BR business case, the following set of process-specific requirements, organized according the taxonomy depicted in table 3.1, has been

drafted (only the requirements categories affected by new process-specific inputs are considered):

- **Functionality**: the SLPS solution should derive actions to improve machine tool efficiency in terms of energy consumption and maintenance scheduling activities taking into account both actual manufacturing production line context and past context. To do this, explicit representative models of production machinery extracted from contextual information are needed. The SLPS solution should communicate with machine tool control, i.e. provide input to machine tool control and/or extract historical machine data, using an available Generic Server Data (GSD) based on the *OPC Unified Architecture*.
- **Maintainability**: The behavior of SLPS solution has to be deterministic and traceable, so that the results of all processes relevant for production are reproducible.

An architectural overview of the BR business case together with SLPS solution is depicted in figure 3.2.

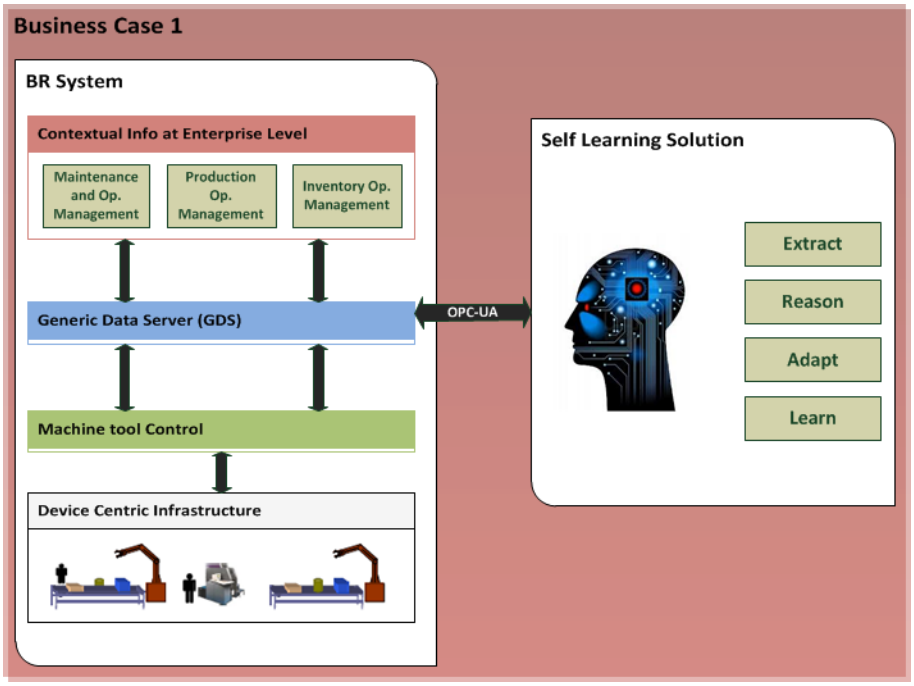


Figure 3.2: Architectural overview SLPS solution integration with business case 1

3.1.2.2 Desma - Business Case 2: Optimization of Manufacturing process parameters for the Shoe Industry

The manufacturing processes of today are caught between the growing needs for high quality, high process safety and minimal manufacturing costs. In order to meet these demands manufacturing process setting parameters have to be chosen in the best way

possible [Rao, 2010]. The selection of optimum process parameters ensure best product quality while reducing manufacturing costs in terms of failures.

Therefore, the second business case reports the application of SL solutions to extend current system monitoring and control skills. The objective is **the recognition of anomalous situations, that potentially may cause a line stopping or degradation of product quality, from variations in process parameters and react to this variations providing adjusted process parameters set. All the adjusted process parameters have to be presented to human operator for final decision**.

According to this objective, this business case involved to equip of the current production system with a number of additional sensors to allow a better monitoring of status and performance of the line. The SLPS solution will then process the information, coming from the shop-floor machines, and correlate it with the entire lifecycle of the manufacturing production line in order to estimate the process parameters trend in machine/component behavior along time and recognize patterns in their variations.

Finally, Desma intends to apply SLPS solution to three distinct application scenario, respectively: tanks refilling, valve synchronization and cup foam.

Taking into account objectives and goals of Desma business case as well as the different proposed application scenarios, the following set of process-specific requirements, organized according the taxonomy in table 3.1, has been drafted (only the requirements categories affected by new process-specific inputs are considered):

- **Functionality**: the SLPS solution should support monitoring of process parameters from different sources, extract contextual information from production environment and based on it should support adaptation of machine parameters. Specific machine parameters should be adjusted depending on the particular application scenario. Furthermore SL solution should be capable to understand the principles governing the related manufacturing processes.The SLPS solution should communicate with company's production system components for retrieving all the necessary information to allow adaptation.

- **Usability**: The SLPS solution should embed in the all-day working while monitoring and tracing unobtrusively the existing technical environment.

An architectural overview of the Desma business case together with SLPS solution is depicted in figure 3.3.

3.1.2.3 Fastems - Business Case 3: Intelligent Scheduling and Dispatching in Flexible Manufacturing Systems (FMS) for automotive industry

Modelling and Scheduling of Flexible Machine Systems (FMS) has been extensively studied and analyzed by several authors [Stecke, 1985, Jain and Elmaraghy, 1997, Rossi and Dini, 2000, Reyes et al., 2002].

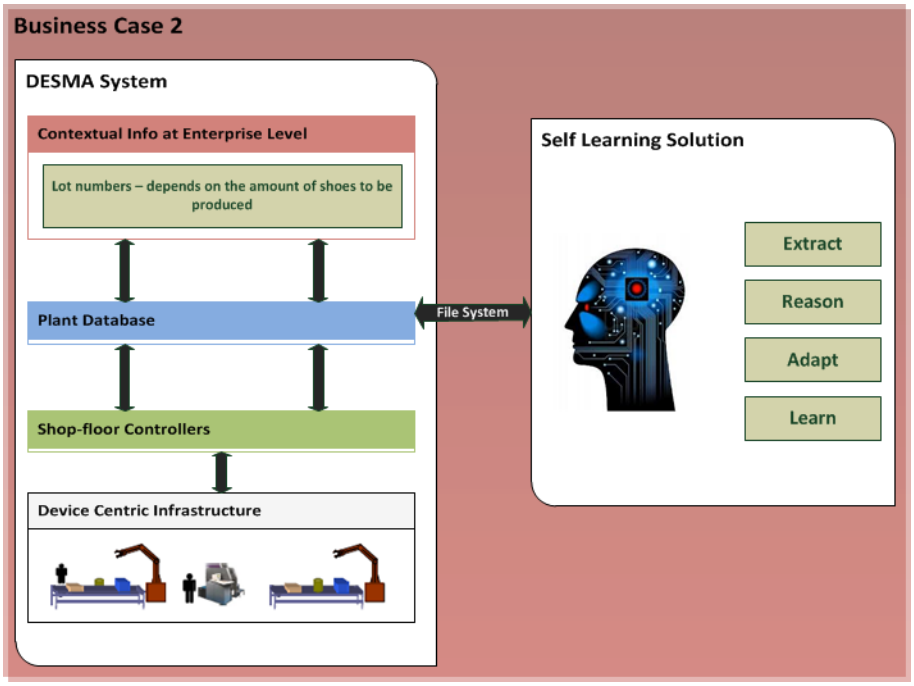


Figure 3.3: Architectural overview SL solution integration with business case 2

A FMS is constituted by several manufacturing high-specialized machines together with automated material handling systems enabling material transport from one machine to another. Each product can be manufactured via one of several available routes and tools implying that an high level controller have to decide how to allocate and/or control the available resources according to the production planning (scheduling and dispatching) to optimize determinate performance criteria. The optimum management of FMS resources is fundamental in the field of industrial production for decreasing production costs. As stated in [Rossi and Dini, 2000], the performance of a FMS not supported by an efficient scheduling and dispatching of the resources drastically reduce the advantages derived from its flexibility. Furthermore, the intrinsic heterogeneity of a FMS makes the identification and definition of a strategy for their management more difficult.

The third business case at Fastems is placed in this context, and grounds on the deeper use of SL solution to **improve and/or advance general performance of FMS cell by optimizing the machine scheduling and dispatching model according to the current context**.

In real manufacturing production systems the availability of the resources is strongly dynamic along time. Uncertainties in the production environment have to be considered: delays in part availability, machine breakdowns, tool failures, variable markets demands and so on are only few aspects that make impossible the use of unreconfigurable scheduling and dispatching plans. These factors make the dynamic and/or reactive scheduling essential.

Fastems intends to apply SL solution together with the existing monitoring and control platform for gathering all the necessary information about the manufacturing production process as well as input from human expert experience concerning the optimization criteria. The SLPS solution will process all the amount of gathered data to dynamically adapt the scheduling plans for avoiding loading stations starvation while improving maximum machines utilization.

Finally, taking into account objectives and goals of Fastems business case, the following set of process-specific requirements, organized according the taxonomy in table 3.1, has been drafted (agin, only the requirements categories affected by new process-specific inputs are considered):

- **Functionality**: the SLPS solution should support extract contextual information from flexible manufacturing cell and based on it should propose to operator the best rule to apply to the given the current context. Furthermore SLPS solution should be capable to classify flexible manufacturing cell operative contexts in order to select the best scheduling/dispatching rule. The selected rule should be presented to a human operator for final validation. The SL solution should communicate with FMS using an already existent Web Services infrastructure.
- **Efficiency**: The SLPS solution should be able to operate in Windows based PC environment and to respond to a context change in less than four seconds.

An architectural overview of the Fastems business case together with SLPS solution is depicted in figure 3.4.

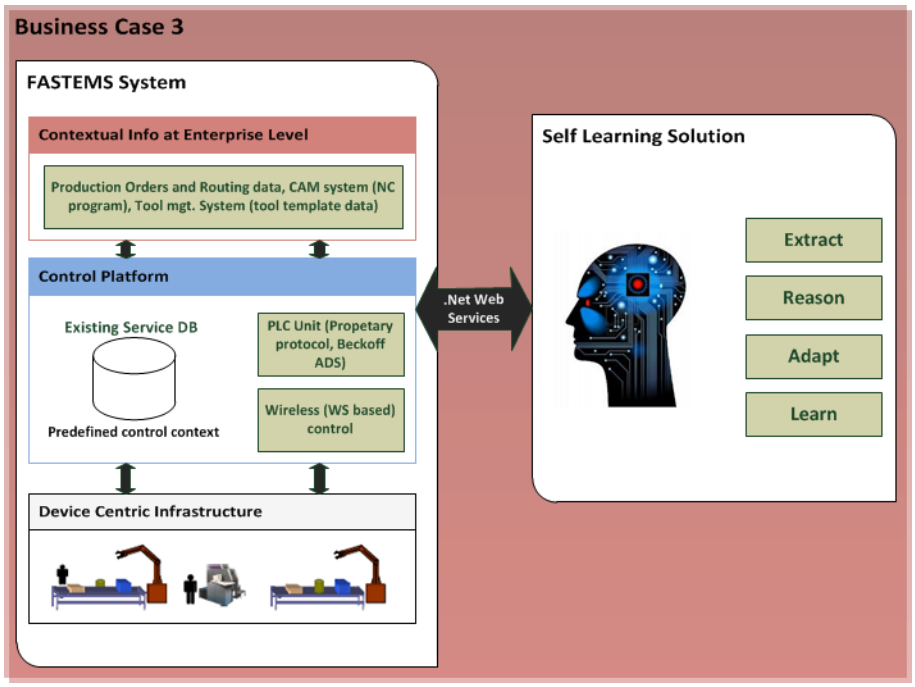


Figure 3.4: Architectural overview SL solution integration with business case 3

3.2 Generic Architecture Overview

The requirements analysis derives the main features and functionalities for the overall SLPS architecture (see Fig. 3.5) allowing to meet industry specific needs while enhancing manufacturing production system agility.

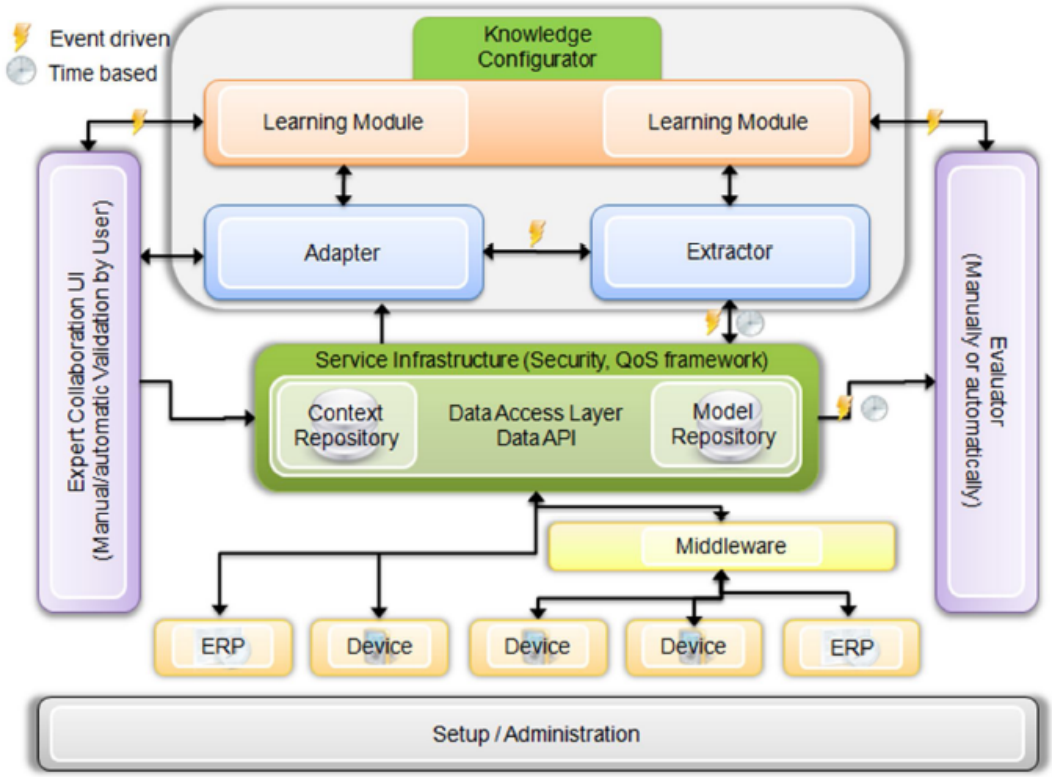


Figure 3.5: Self-Learning Architectural overview [Uddin et al., 2011]

The components of SLPS architecture are the following:

- ***Extractor***: is responsible for dynamically extracting, processing, filtering and finally storing actual manufacturing production process context.

- ***Adapter***: is the active/reactive component responsible for processing and filtering contextual data and adapting manufacturing process parameters, i.e. control parameters, maintenance/energy plans and/or scheduling/dispatching execution to face changes in process.

- ***Learning Module***: encapsulate machine learning algorithms and process explicit models to learn relying on data mining and operator's feedback. Furthermore, is responsible to update process explicit models after human expert validation and/or feedback.

- ***Expert Collaboration UI***: is responsible to show *Adaptation proposals* about adjusted parameters and processes allowing human expert validation, i.e. the human expert can accept or reject the *Adapter* suggestion. The result of validation is then sent to the *Adapter* and *Learning Module*.

- ***Knowledge Configurator***: is responsible for allowing manual configuration of user inputs to the *Learning Module* in order to enhance reliability and system performance.

- ***Evaluator***: Performance of the adaptation and context extraction as well as process models generalization capability are measured by the *evaluator*. Evaluation results are then sent to the *Expert Collaboration UI* to be presented to the human expert to assist him during the validation process.

- ***Data Access Layer***: wide range of data is available in *Data Access Layer* from the plant floor infrastructure. The *Model Repository* contains ontology based plant specific models for equipment, production processes and products. The models are shared by different software components at run time. The *Context/Monitoring Repository* allows update and storage of extracted/processed contextual information for later retrieval. The *Adaptation Repository* allows update and storage of all the information generated by the SLPS *Adapter*. Information flow among the modules is event driven in some cases and time based in other cases.

- ***Service Infrastructure***: is responsible to ensure secure and reliable information flow between the SLPS solution and the existent manufacturing process equipments. The information flow is bi-directional allowing both the collection of relevant contextual information from the process and communication of adjusted manufacturing process parameters to the monitoring and control system.

- ***Middleware***: Information from ERP level, devices or plant data servers are brought to *Data Access Layer* directly or via middleware depending on plant specific equipment and communication protocols.

3.2.1 Generic Architecture Description

The primary functionalities and interactions of the different components included within the SLPS architecture are described in this section.

The entire architecture is evolved around the main core requirements of "**sensing**", "**extracting**", "**reasoning**", "**adapting**" and "**learning**". According to these functionalities and/or requirements, the reference architecture is constituted by two central components: the *Extractor* and the *Adapter*. The *Extractor* acts as an observer of manufacturing process during its operation, while the *Adapter* acts as a doer allowing adaptation of manufacturing process parameters along time. Therefore, these two components together are

responsible for identifying the current context under which the production system is operating (*Extractor*) and adapt the production system behavior, i.e. manufacturing process parameters, in order to improve its performance in face of that contextual change (*Adapter*). The result of the context extraction process will be a standardized meta-model gathering all context relevant information obtained by monitoring process machines and devices. This standardized meta-model will be used by the *Adapter* to start an evaluation process for improving production system productivity, efficiency and performance. Finally, the outcome of the adaptation activity, the *Adaptation* itself, is exposed to the system expert through the *Expert Collaboration UI* for a system expert evaluation.

Since the system response must take into account not only the particular context, but most important, the entire lifecycle behavior of both system and expert, a *Learning module* has been provided, containing a set of machine learning algorithms capable for extracting patterns and regularities from gathered contextual data and operator decisions along time. All processed data and knowledge generated are stored in *Data Access Layer* repositories for continuous evolution. These components allow both *Extractor* and *Adapter* to access them when they need further information about current context and/or results of adaptation activities.

The reference SLPS architecture has been designed following a modular and abstract approach in order to remain hardware-independent and still compliant with a wide range of application domains.

3.2.2 Self-Learning Adapter

As stated in subsection 3.2.1, the *Adapter* component acts as a doer within the SLPS architecture meaning that it has to provide manufacturing process parameters adaptations whenever changes in process operative conditions (process is acting in a new operative context) are detected and notified by the *Extractor*. Therefore, it is responsible for updating system behavior (locally and/or globally) in response to a change of context in the environment. Moreover, adaptations of parameters have to consider not only changes in operative contexts but also system evolution along its entire lifecycle in order to identify the best adaptation to employ to handle the new reality.

The *Adapter* is presented by describing its behavior on top of the proposed architecture, especially showing its main modules and the interactions between them during the adaptation process.

The issues introduced in the previous sections have been deeply studied during the first stages of the Self-Learning project, which resulted in a set of requirements and functionalities expected to be supported by the *Adapter* and resumed as follow:

- React to a change of context and provide a suitable adaptation proposal to be validated by the system expert.
- Employ the Learning module as a mean to process large amounts of data concerning a particular context and identify the fittest adaptation proposal to be presented

to the system expert.

- Detect expert decisions about system adaptation result and deploy the validated adaptation into the system.
- Manage Adaptation Repository ensuring that each adaptation process is stored for future use and analysis.
- Proactively process existing data during system idle time and/or when a model is considered out-dated to update its core knowledge about system evolution.

3.2.2.1 Adapter Generic Reference Architecture

The generic *Adapter* architecture is shown in figure 3.6. The core task-oriented components of the proposed architecture are the following:

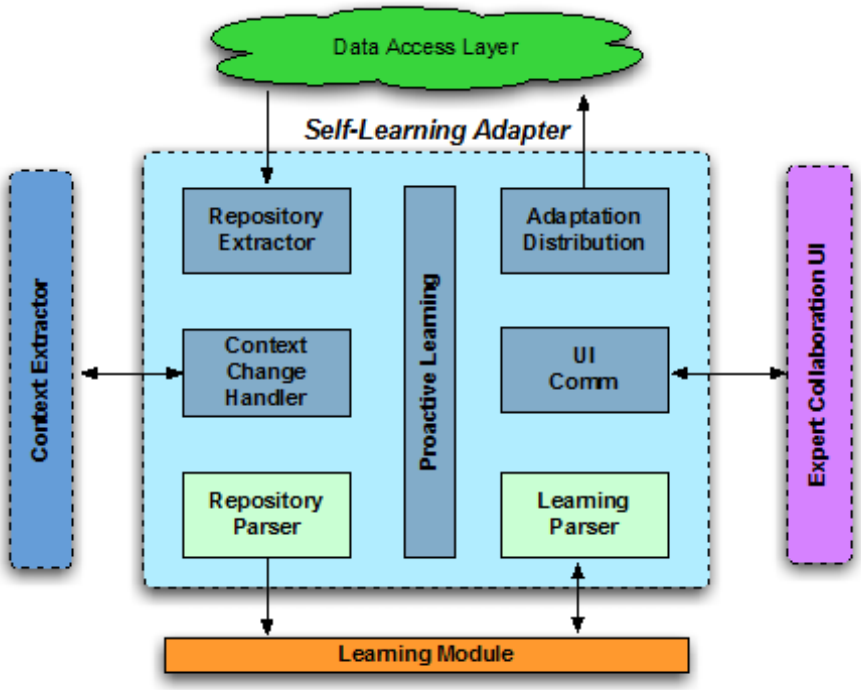


Figure 3.6: Adapter architectural overview

- **Context Change Handler**: responsible for asynchronously handle notification events sent by the *Extractor*, whenever a change in context is detected. These events will be the trigger of an adaptation process.
- **Repository Extractor**: responsible for retrieving the necessary information from the *Data Access Layer* repositories related to the current context change. The retrieved data set includes all the information necessary to support the adaptation

process that will, in turn, determine the appropriate adaptation proposal, i.e. machines/processes parameters and/or configurations adaptation to the new context.

- **Repository Parser**: the data set retrieved from *Data Access Layer* repositories contains raw information that needs to be arranged in particular way in order to be properly processed by the *Learning Module*. In summary, the *Repository Parser* creates a generic data structure that will serve as input for the *Learning Module*.

- **Learning Parser**: Similarly to the *Repository Parser*, this component will acquire the result of the *Learning Module* reasoning task and parse it to create a generic data object (*Adaptation*), which includes all the information needed by the system expert for validation issues. Furthermore, it is also responsible for receiving a complete *Adaptation* (including the proposal and result of the system expert validation). This information is crucial to support the accuracy of future adaptation proposals.

- **UI Comm**: handles the interaction between the *Adapter* and the *Expert Collaboration User Interface (UI)* providing a communication channel between the system expert and the SLPS deployment. This component is also responsible for informing both the *UI* whenever a new adaptation proposal is ready and for detecting/retrieving an adaptation that was entered to the system through the *Expert Collaboration UI*.

- **Adaptation Distribution**: responsible for distributing an *Adaptation* object instance along the Self-Learning environment after it was transferred into the real system. It will store the current *Adaptation* instance in the *Adaptation Repository* and it will inform *Context Extractor* that an adaptation was done in the system.

- **Proactive Learning**: This module embodies the proactive behavior of the *Adapter* component by proactively performing the relearning of the models based on existing context data. This task can be event-triggered or a cyclic task depending on initial configuration. The major goal is to improve future adaptation proposals and exploit system idle times to run time and processor consuming learning tasks.

Each core component of the *Adapter* plays a different role during the process of collaboratively providing adaptation proposals concerning adjusted manufacturing process parameters to properly face the current context. Moreover, the specifications and design of *Adapter* reference architecture has followed a modular and generic approach to remain compliant with each business case details, i.e. despite business cases *minutiae* are specific to each application scenario the *Adapter* reference architecture is generic enough to cope with the three of them without any modification in its structure.

Since the *Adapter* is simply one brick of the overall infrastructure, it needs to interact with other surrounding modules to entirely fulfill its goals. This way the *Adapter* will interact with the following infrastructure modules: *Extractor, Learning Module, Expert Collaboration UI* and *Data Access Layer*. To better clarify the *Adapter* role within the SLPS

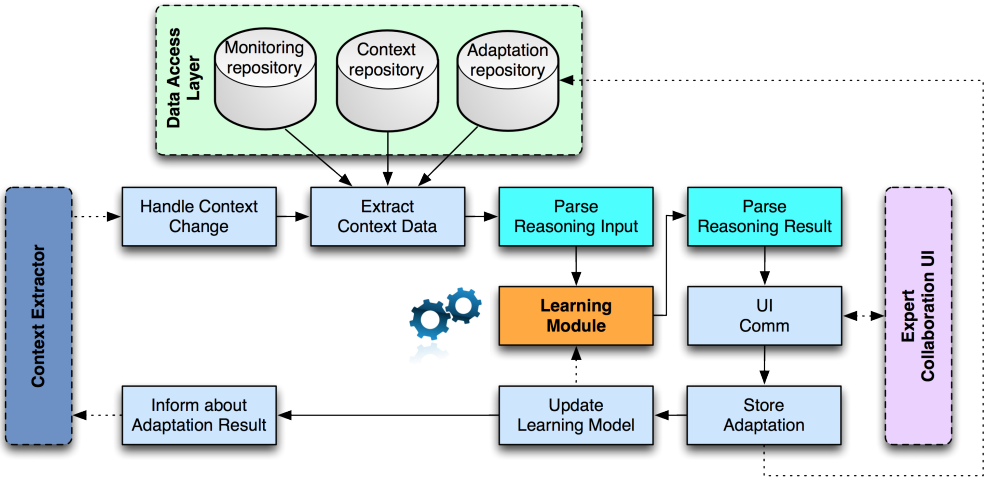


Figure 3.7: Adaptation process overview

environment and thus the interactions with the others modules as well as the role of its own constituent components, an adaptation process will be explained.

3.2.2.2 Adaptation Process

The adaptation process (see figure 3.7) refers to the execution of a sequence of tasks for evaluating production system parametrization consistency, which are performed every time the *Extractor* notifies the *Adapter* about a change of context. This notification represents the trigger that will drive the SLPS to adapt itself to the current context.

After being notified about a change of context, the first thing the *Adapter* do is to retrieve all the available information related to this new context. This task is performed by the *Repository Extractor*, which retrieves, from the *Data Access Layer*, all the according datasets and models for the current context (monitoring dataset). This collection of data is then transferred to the *Repository Parser* component responsible to transform it into a generic data structure, (*ReasoningInput*), that structures the retrieved monitoring dataset. The *ReasoningInput* will be used as input for the *Learning Module* whenever there is a need to perform a reasoning task. A reasoning task will exploit supervised machine learning classification techniques that, based on an explicit learning model comprising the entire lifecycle behavior of the system, will breed new production system parametrization proposals. Subsequently, the *Learning Module* will deliver the reasoning task result to the *Learning Parser* component and create the *Adaptation* object instance. Specifically, at this moment, this object instance will include the current context dataset that has triggered the adaptation process, the adaptation proposal containing the result of the reasoning task, which is going to be shown to the system expert, and finally the adaptation result which is going to be filled with system expert final decision. This way, the *Adaptation* object instance will be transmitted through the *Comm UI* component to the *Expert Collaboration UI* that waits for the system expert to validate, modify or refuse the original

adaptation proposal. The system expert decision will be stored in the *Adaptation* object instance as adaptation result, and at the same time this adaptation result will be transferred to the real production system equipment. Finally, the *Adaptation* object instance will be distributed within the SLPS environment, i.e. notify the *Extractor* concerning the end of the adaptation process, save the *Adaptation* into the according repository and invoke the *Learning Module* service to update existing learning models with the related context information and adaptation result. Since supervised machine learning techniques are used to adapt system behavior during its lifecycle, an externally supplied set of context instances are required to build a learning model for the particular application scenario. Once a learning model is available, an inductive machine learning process can be carry out representing the process of learning a set of rules from instances (examples in a training set), creating a classifier/estimator that can be used to generalize from new instances [Kotsiantis et al., 2007]. During the classification/regression task the choice of the specific learning algorithm is a critical step. The learning algorithm choice has been realized considering prediction accuracy and/or relative error. In this scenario several statistical comparisons were conducted between the learning algorithms coming to a final well suited solution based on the nature of each application scenario.

3.2.2.3 Proactive behavior

The envisioned *Adapter* reference architecture points out a purely reactive behavior triggered by the *Extractor* whenever a change in production context is detected, however some kind of proactivity has been embodied in *Adapter* architecture for increasing its autonomy, saving time and resources during system run-time while allowing configuration of learning models evolution trend along time. Thus, the *Adapter* is, then, capable of monitoring its own state during system operation in order to identify instants in time to proactively launch new learning tasks (Proactive Learning). Therefore, when a learning task is launched, i.e. whenever a learn command is sent to the *Learning Module*, a new explicit learning model of the process referring to all the stored contexts information is inferred and a *Cross Validation* [Kohavi et al., 1995] is also performed to analyze the capability of the model to generalize in face of an independent dataset. The result of the cross validation is then stored into an appropriate repository that can be queried by the user for retrieving statistical information useful for him to assess the quality of the adaptation proposals. Since the learning task is a time consuming activity, the proactive behaviour should optimize this process identifying inactivity periods as well as obsolete learning models.

The figure 3.8 presents an overview of the *Adapter* proactive behavior, its interactions with the *Learning Module* and its role during the *Adaptation process*.

Therefore, during production system operation, the *Adapter* will verify the number of performed adaptation processes and the elapsed time since last adaptation for detecting when a model can be considered out-dated and/or when the system is idle.

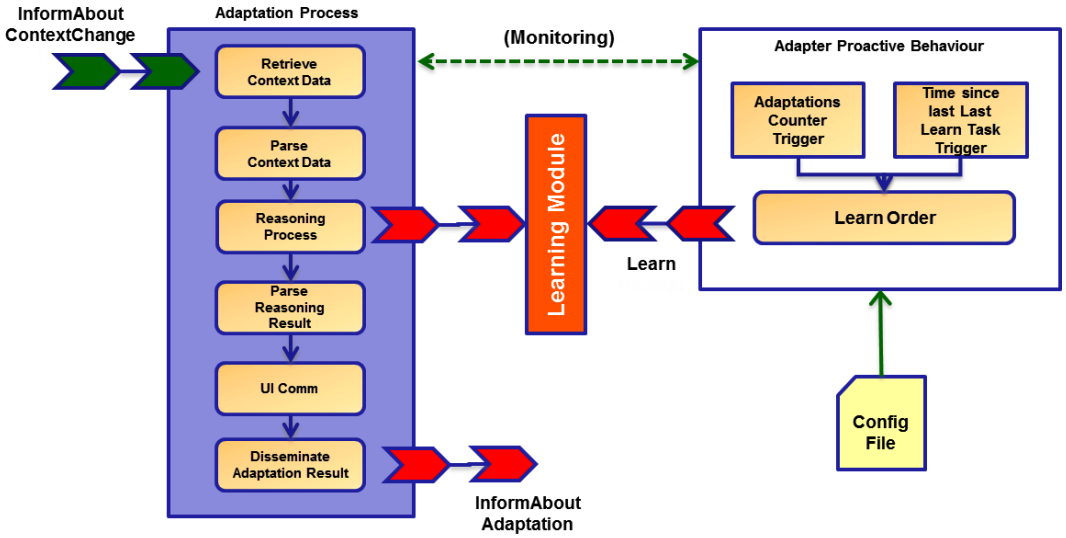


Figure 3.8: *Adapter* proactive behavior

3.2.3 Self-Learning Learning Module

The *Learning Module* represents the reasoning unit of the *Adapter* component and is responsible for computing the overall production system parameters whenever a new context is detected by the *Extractor*. It comprises machine learning algorithms and explicit models of the manufacturing process. The learning algorithms are responsible for both processing the huge amount of manufacturing data gathered during production activities and construct an explicit model of the manufacturing process rely on hidden relationships between data. Furthermore, the explicit models are attached to the related algorithm and the mapping between models and algorithms is managed by a dedicated component named *Models Manager*. Thus, the *Models Manager* is responsible for managing the models, i.e. for both upload a model whenever is needed and update the models with new entries from the detected contexts. The figure 3.9 shows the constituent architecture of the *Learning Module*.

The *Learning Module* is triggered by the *Adapter* component through a standard interface named *Learning Service* during the adaptation process. To better understand the *Adapter* behaviour inside the overall SLPS architecture an overview of the collaboration with the *Learning Module* is depicted in figure 3.10. The interaction between the *Adapter* and the *Learning Module* is defined as *Learning process*.

The *Learning process* is started during an *Adaptation process* and consists of of two main tasks executed at different point in time, namely: processing information from the *Adapter* and updating the models of the process according to the system expert final validation. The former consists of a set of activities allowing both the processing of the *Reasoning input* (a standard data structure from the *Repository Parser* containing the actual context in a well-defined form) using machine learning algorithms in order to calculate the best

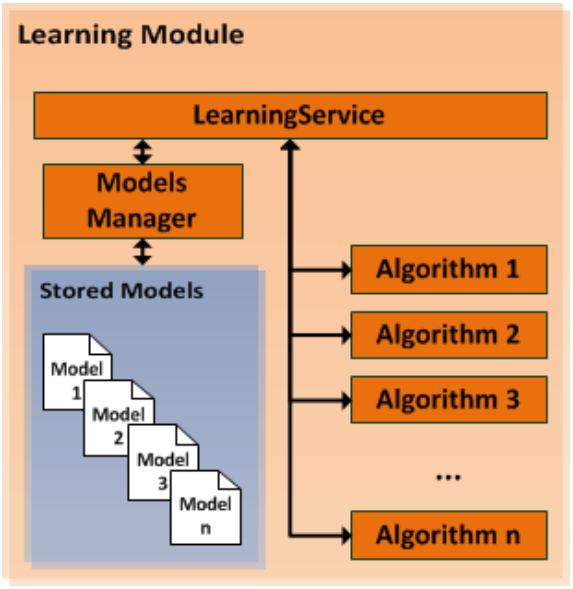
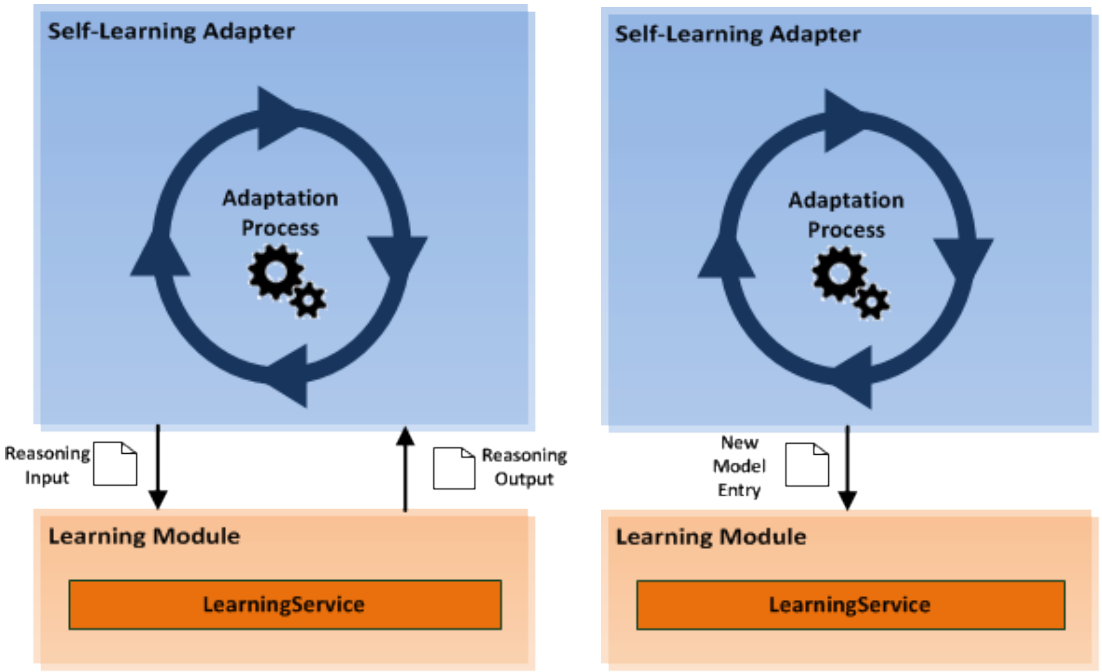


Figure 3.9: *Learning Module* internal Architecture

suitable set of production systems parameters to face the current contexts and the communication of the result, encapsulated into a standard data structure named *Reasoning Output*, to the *Adapter*; while the latter enables the updating of the models of the process attached to each algorithm if new entries from system expert are available, i.e. whenever the system expert change the adaptation proposal.



(a) (b)

Figure 3.10: *Learning module* - interactions with *Adapter*. (a) Processing contextual information. (b) Updating models of the process.

Finally, the SLPS solution is intended to be an evolvable architecture capable to acquire new knowledge during its lifecycle, In this direction, the description of the adaptation and learning processes emphasize that SLPS *Adapter* together with the *Learning Module* behaves as a closed-loop feedback system in which the information about the past is continually used to make prediction about the present or future and, above all, any correction made by the system expert is stored to be considered for future adaptations as shown in figure 3.11, where the numbers display the sequence of the adaptation and learning processes.

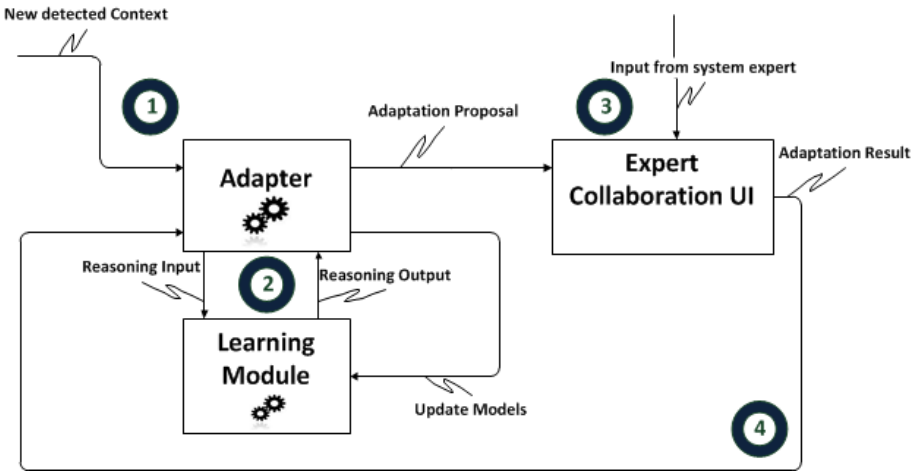


Figure 3.11: Feedback between the SLPS platform and the system expert

After a general introduction about the behaviour of the designed SLPS components and their related processes necessary to frame their role inside the platform, the following subsections are intended to present in a more comprehensive and detailed way all the interactions among these components. Since the focus of the current dissertation is the *Adapter* with its own *Learning Module* the attention is given to them, pointing out the information exchange with the other components of the SLPS platform as well as the internal information exchange between their own components during system operations. The interactions are shown using a formal representation: Unified Modelling Language (UML) Sequence Diagrams.

3.2.4 Adapter and Learning Module: interactions within SLPS architecture

The figure 3.12 shows a typical information exchange between the SLPS *Extractor*, *Adapter*, *Learning Module* and *Expert UI* whenever a new context is detected in manufacturing production system. The *Extractor* informs the *Adapter* of a new available context, which in turn starts the adaptation process aiming to calculate the best suitable manufacturing production system parameters to face the new reality in which the system is operating. The new values of the parameters are calculated exploiting machine learning techniques included into the *Learning Module*. The final result of the adaptation process is then sent to the *Expert UI* as an adaptation proposal for a further validation by a user expert. Finally,

the result of the validation is fed back into the SLPS platform.

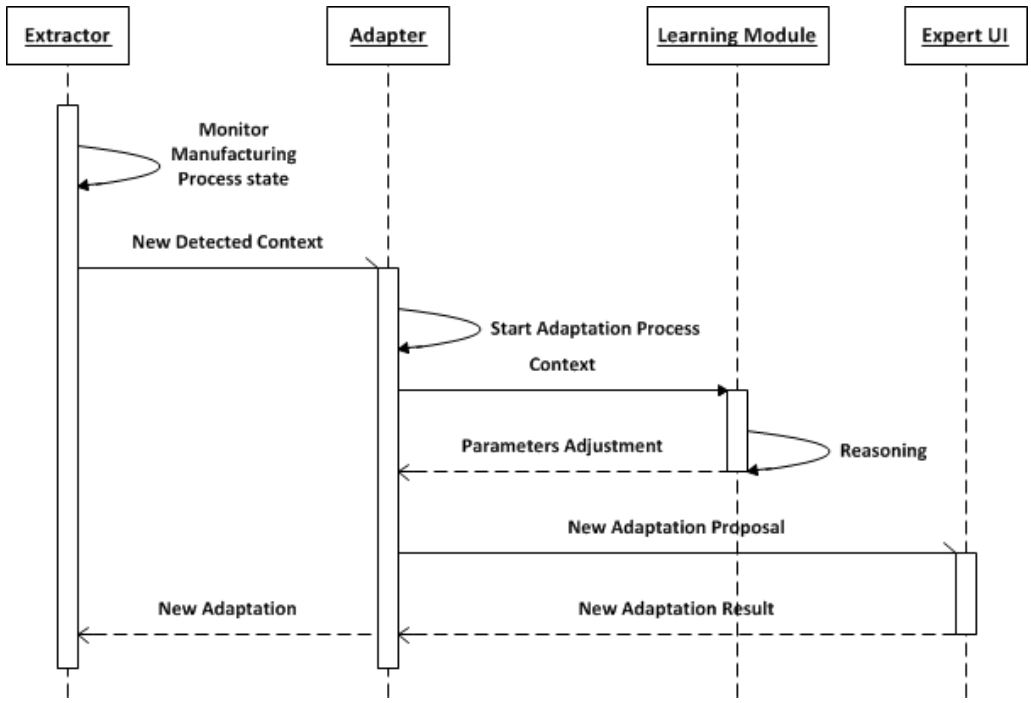


Figure 3.12: Execution of an adaptation process (*Extractor*, *Adapter* and *Expert UI* interactions)

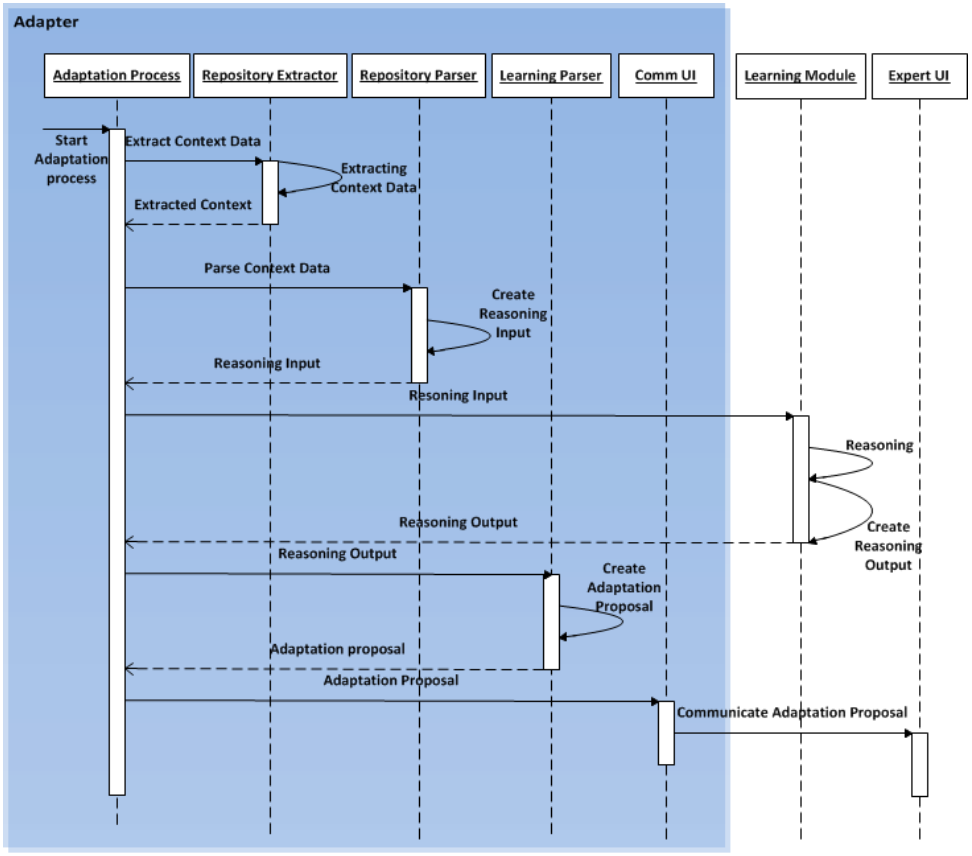
3.2.4.1 Detailed interactions between Adapter internal components

The process of obtaining an adaptation proposal when a change in context is detected relies on several interactions between all the components inside the *Adapter*. This process can be logically separated into two distinct phases. The first one start when the notification from the *Extractor* about a new detected context is received and terminates when the adaptation proposal is communicated to the *Expert UI* for validation issues. Furthermore, the second phase starts when the validated adaptation, i.e. the adaptation containing the adjusted parameters validated by the user expert is received from the *Expert UI*, and terminates with a notification to the *Extractor*, informing about the end of the adaptation process, is sent.

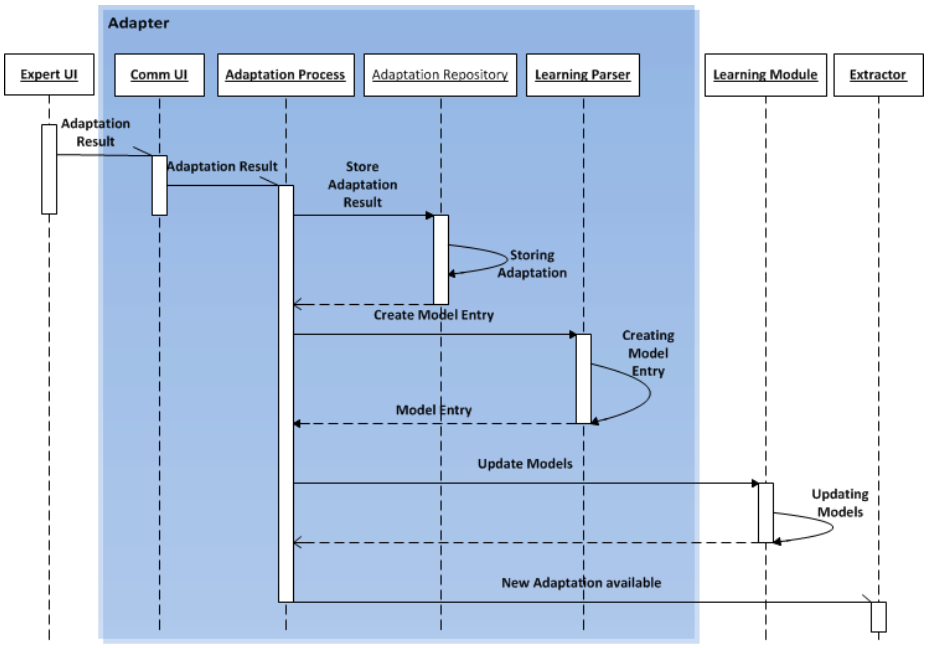
The figures 3.13(a) and 3.13(b) show these two phases.

3.2.4.2 Detailed interactions between Learning Module internal components

Whenever an adaptation process is started there is the need for computing the adjustment of manufacturing production process parameters. The evaluation process of the parameters is accomplished by the *Adapter* resorting to a set of resources and functionalities provided by the *Learning Module*. Accordingly, the *Adapter* will send all the necessary information about the actual context to the *Learning Module* that in turn will return the set of adjusted parameters values. The details of the information exchange between



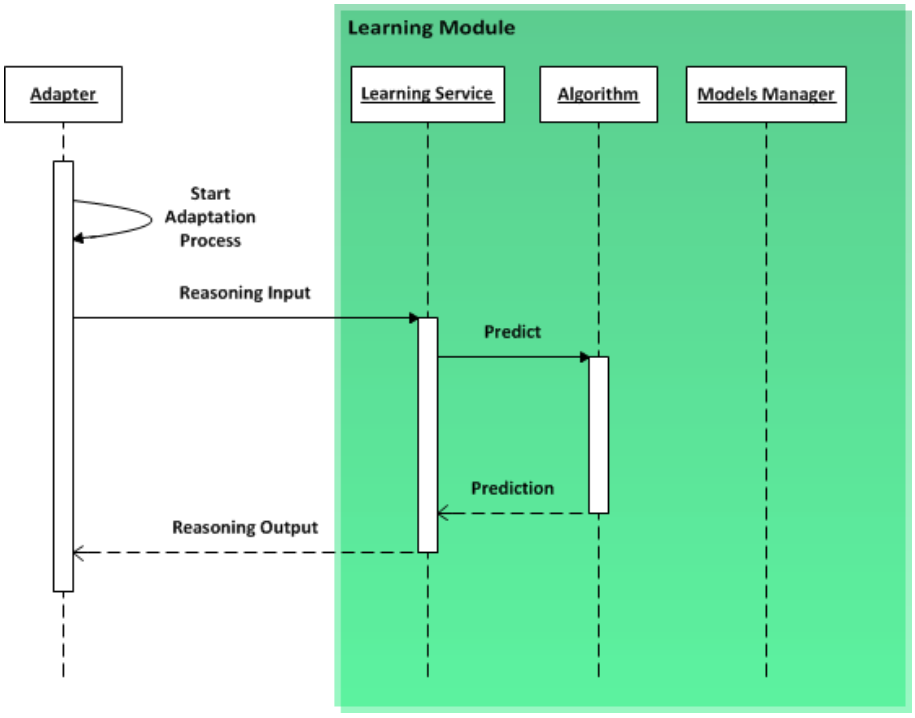
(a)



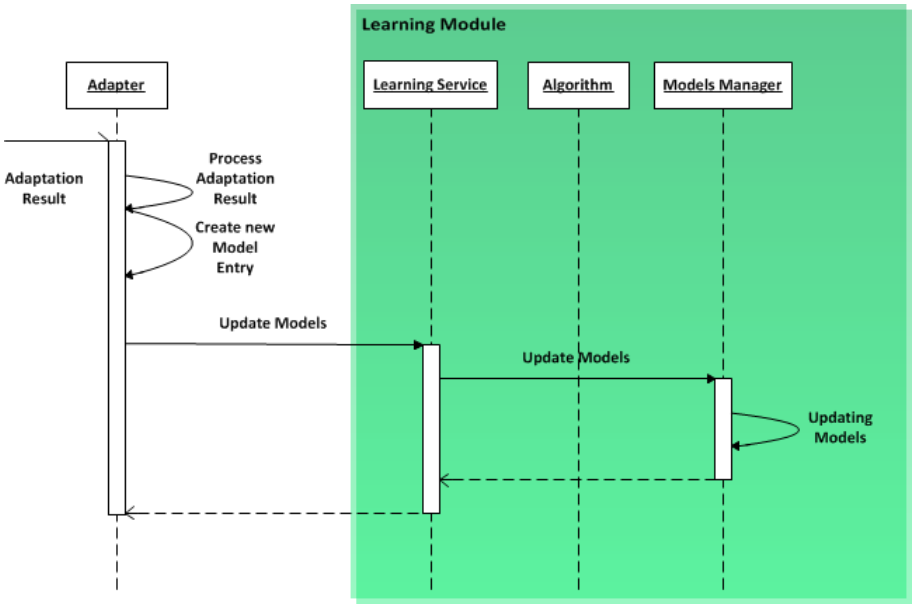
(b)

Figure 3.13: Information exchange inside SLPS *Adapter*. (a) Start adaptation process and show adaptation proposal. (b) Receive the adaptation result and update models.

these components during the reasoning task is shown in figure 3.14(a). Moreover, when the adaptation result from the *Expert UI* is fed back to the SLPS platform, the *Adapter* is responsible for communicating the validated result to the *Learning Module* to allow the updating of the explicit learning models of the process. The figure 3.14(b) shows the detail of this interaction.



(a)



(b)

Figure 3.14: Information exchange inside SLPS *Learning Module*. (a) Start adaptation process and reason on a new context. (b) Receive the adaptation result and update models.

3.2.5 Use Case diagram

The SLPS methodology and concept relies on learning activity to improve future adaptations and system performance along time. The foundation of the learning mechanism is a strong interaction between SLPS platform and user expert, considering the expert background and technical knowledge as source of learning. For this reason the SLPS platform provides an *Expert UI* to ensure interactions between user and SLPS and above all to allow the insertion of new knowledge into the platform. The figure 3.15 shows the Use-Case diagram summarizing the principal interactions between the SLPS and the user of the system.

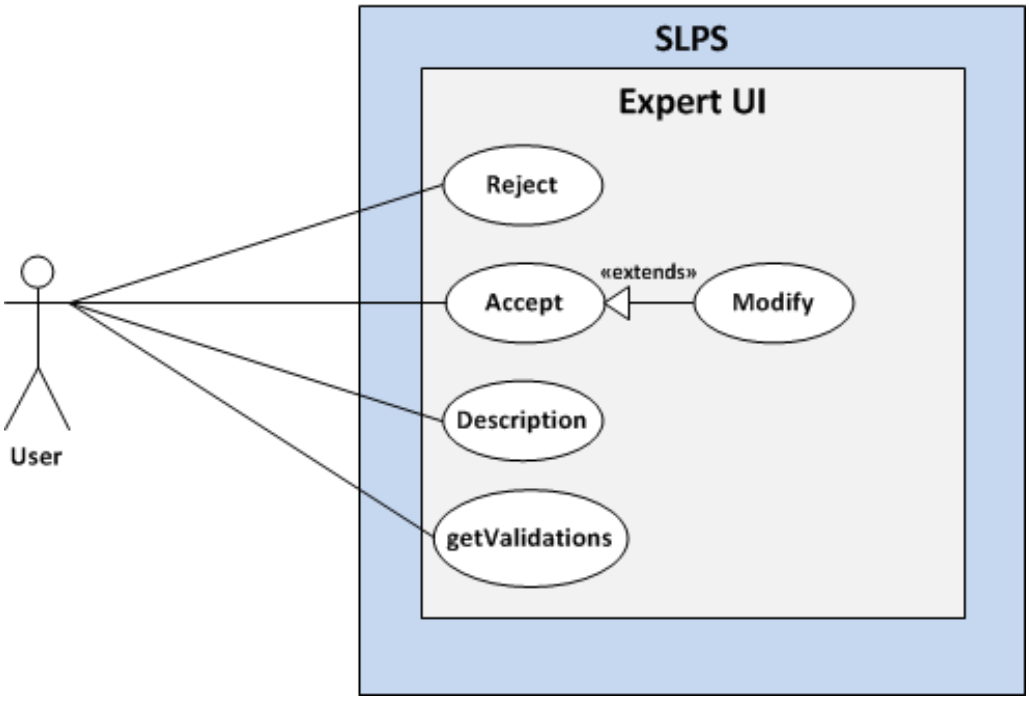


Figure 3.15: SLPS Use-Case diagram

Looking for the figure 3.15, the following use-cases have been considered:

- **Reject**: the user can simply reject the SLPS adaptation proposal without sending any feedback to the system.
- **Accept**: the user can accept the SLPS adaptation proposal with or without making modifications. In this case the modified adaptation is sent back to the SLPS platform that will use it to update the learning models of the process.
- **Description**: the user can check, whenever it is necessary, all the values of the process parameters that have gave rise to the adaptation proposal. In this way the user expert will be able to contextualize the adaptation proposal.
- **getValidations**: the user can verify the performance of the SLPS platform and its

evolution along time by retrieving the results of the *X-Cross* validations run according to a preconfigured allowing, in such a way, the user to have an idea on how good can be the adaptation proposal.

These are the main functionalities that the SLPS platform has to provide to the user, however new functionalities can be added according to the particular application scenario in which SLPS platform is integrated without loosing system core genericity.

4

Prototype and Validation

This chapter aims to discuss all the aspects related to the implementation of the SLPS *Adapter* and *Learning Module* core components according to the architectural guidelines presented in the chapter 3 and detailing all the development tools used. Moreover, a SLPS prototype, including all the implemented components, is presented. The SLPS prototype has been applied to three real industrial environments, provided by the three industrial partners of the Self-Learning project, and integrated in their manufacturing production processes to validate and test the concept, the feasibility and reliability of the SLPS platform as a whole, as well as, its performance in terms of adaptability, evolution and learning capability along time. Thereby different application scenarios has been considered providing a wide variety of manufacturing experimental setups and problems to face, namely: dynamic scheduling and dispatching of resources in FMS according to the production orders, manufacturing process parameters optimization to improve product quality and reduce line stopping and, finally, enhance energy efficiency during production activities. The distinct application scenarios are described in the following sections together with the steps of the methodology required to configure and operate the SLPS platform, i.e. the configuration and needs of the *Adapter* and *Learning Module* components to deal with each one of the considered application scenarios.

4.1 Development Software Tools

The implementation of the key elements of the SLPS architecture is fundamental for proving that proposed methodology and concept are feasible. By developing a prototype, the feasibility of the SLPS concept and methodology are proved, which enables the achievement of results in the other dimensions of validation namely: reliability, effectiveness,

adaptability, evolution and learning capability along time. Thereby, for the implementation of the SLPS prototype several development tools and Integrated Development Environment (IDE) have been used. The open-source Netbeans IDE has been used for the development and the orchestration of all SLPS components. All the software components are based on Java and employed as a whole in the same design and development environment.

The implemented SLPS solution uses the Apache CXF framework as the foundation for the SOA modules and orchestration/composition, which provides web-service based methods and transportation channels in order to realize inter-service communication. The Apache CXF is an open source services framework allowing the development of services using frontend programming Application Programming Interfaces (APIs) like JAX-WS and JAX-RS. These services can speak several message protocols and works over several transports protocols, however SLPS services rely on SOAP and HTTP.

The learning capabilities needed to support the improvement of the SLPS *Adapter* adaptations during the system lifecycle are implemented using the RapidMiner API which provides a wide range of multi-purpose algorithms and operators tools for knowledge management, machine learning and data mining applications as well as statistical performance validation.

In addition, storage capabilities are needed to ensure and support access to all the context data and adaptation generated during SLPS operations. The h2 database engine is used, providing an open source pure Java based relational database management system (DBMS) that can be embedded in any Java application and managed, i.e. connection to the database and queries execution, through the Java Database Connectivity (JDBC) API.

Since the development of the SLPS prototype is a joint effort between the consortium's members, the Apache Maven and Subversion tools, included into NetBeans IDE, have been also employed. The former allows to manage the SLPS software project creating a standardized way to build it and reducing, in such a way, version conflicts while allowing JARs sharing across several projects and improving the comprehension about the complete state of a development effort. The latter is an open source software versioning and revision control system used to maintain current and historical versions of files during the development process. In the same direction, the MantisBT, a web-based bug tracking system, has been used during the SLPS prototype implementation phase in order to easily and quickly track software defects and bugs.

Finally, the SLPS prototype has to be applied to distinct application scenarios characterized by different input and output variables. In order to do this, some configurations data are needed to frame the application context. This configuration data are typically stored into XML files and the Simple framework is used to provide a XML framework that enables rapid development of XML configuration and communication systems.

The development software tools used, together with their version, link and name of the task they are being used for are listed in table 4.1.

Development Software Tools			
Functionality	**Software**	**Version**	**Link**
Programming Language	Java	1.6.0_xx	http://www.java.com
IDE	NetBeans	7.0.1	http://netbeans.org
Uniform Build System	Apache Maven	3.0.3	http://maven.apache.org/
Version Control	Apache Subversion	1.7.0	http://subversion.apache.org/
Feature and Bug Tracking	MantisBT	1.1.1	http://www.mantisbt.org/
XML Configuration Wrapper	Simple	2.3.4	http://simple.source.forge.net/
Data Mining and Machine Learning API	RapidMiner	4.6	http://rapid-i.com/
Web Application Framework	Apache CXF	2.5.1	http://cxf.apache.org/
Database Management System	H2	1.3.162	http://www.h2database.com/

Table 4.1: Overview of used development software tools

4.2 Prototype Description

This section details the implementation of the SLPS prototype focusing on the SLPS *Adapter* component and its related *Learning Module*.

4.2.1 Common implementation details

4.2.1.1 Self-Learning Services

The Self-Learning projects provides a SOA infrastructure for implementation of services for SLPS platform. The services constitute the basic communication/integration mechanism between the SLPS components, as well as, between the SLPS platform and the other enterprise software applications. Therefore, all the generated data during SLPS prototype lifecycle can be accessed by invoking the available services. The services implemented under the SLPS context has to met two different requirements, namely:

- **Interface**: a Java interface has to be specified, containing all the methods the implementing service will provide.
- **Implementation**: an actual implementation (Java class) that implements the specified interface has to be provided.

To do that, the standard *ISelfLearningService* interface has been provided and used by the author. Every other service interface, inside the SLPS platform, should implement and eventually extend the standard *ISelfLearningService* interface for the implemented service performing some useful operations. The standard interface for SLPS services, shown in Listing 4.1, provides four methods: *start*, *stop*, *restart* and *ping*. The *start* and *stop* methods are used to start/stop the service respectively, while the *restart* method is

used to restart the service and usually is only required if some configuration has changed. Finally, the *ping* method is used for the determination of the service status, i.e. if pinging is successful the service is running and available inside the platform.

Listing 4.1: ISelfLearningService

```
/**
 * The ISelfLearningService is the basic Interface for any service published
 * within the Self-Learning solution. It provides fundamental functions for
 * starting and stopping of services as well as a mechanism for checking if a
 * service is currently running.
 *
 *
 *
 *
 */@SOAPBinding(style = SOAPBinding.Style.DOCUMENT)
public interface ISelfLearningService {

  @WebMethod(operationName = "startWebService")
  public void start() throws SelfLearningFault;

  @WebMethod(operationName = "stopWebService")
  public void stop() throws SelfLearningFault;

  @WebMethod(operationName = "restartWebService")
  public void restart() throws SelfLearningFault;

  @WebMethod(operationName = "pingWebService")
  public String ping() throws SelfLearningFault;
}
```

As stated in chapter 3, the SLPS *Adapter*, *Context Extractor*, *Expert Collaboration UI* and *Data Access Layer* components have been designed based on service oriented principles to be highly re-usable, self-contained and adaptable to different systems while ensuring interoperability and simplifying, at the same time, the overall SLPS infrastructure by hiding their individual intricacies. As a result, all these components have to implement and extend the *ISelfLearningService* standard interface.

4.2.1.2 Repositories

The SLPS solution makes wide usage of repositories for storing fundamental data during the system lifecycle. As for the repositories related with the SLPS *Adapter*, the *H2* database engine has been used providing a *JDBC API* to easily create, access and query databases. In this context, a generic *IRepository* interface has been developed and used to implement both the *Adaptation Repository* and the *Model Repository*. These repositories are used for storing fundamental data during the adaptation process and other learning activities that will be presented in the following sections. The standard interface for using

the repositories of the SLPS *Adapter* is shown in listing 4.2 and provides the basic operation that each repository should implement and eventually extend if new operations on database are needed.

The *openConnection* and *closeConnection* methods are used to open/close a connection to a database selected according to the *applicationScenario* parameter. The *Insert* method is used to store an *object* into the database, while the *SelectforID*, *SelectforCount* and *SelectBasedOnTime* methods are used to query the database in different way using the *ID*, the *count* and *Time* parameter respectively. The *ID* represents the identifier of the object to retrieve, the *count* specifies the number of objects to retrieve and, finally, the *Time* specifies the time interval to consider when retrieving the objects.

Listing 4.2: IRepository

```
1  public interface IRepository {
2
3      public Connection openConnection(ApplicationScenario applicationScenario);
4
5      public boolean closeConnection();
6
7      public boolean Insert(Object object);
8
9      Object SelectforID(String ID);
10
11     public List<Object> SelectforCount(int count);
12
13     public List<Object> SelectBasedOnTime(TimeFrame Time);
14 }
```

4.2.2 Self-Learning Adapter

4.2.2.1 Behaviour

The SLPS *Adapter* is the component responsible for updating/adapting the manufacturing process behaviour in response to a change of context in the environment detected by the SLPS *Extractor*. Despite the reactivity represents the typical behaviour in SLPS *Adapter* it is not the only one since some kind of proactivity has been embodied to increase its autonomy along time. The UML diagram in figure 4.1 depicts the SLPS *Adapter* overall activities when the SLPS platform is running. These activities are explained with more details in the following sections.

4.2.2.2 Communications

Since the SLPS *Adapter* is one brick of the overall SLPS infrastructure, it needs to be connected to other surrounding SLPS components to entirely fulfill its objective. Therefore, some communication mechanism is needed to allow the SLPS *Adapter* to interact with the SLPS *Extractor*, *Learning Module*,SLPS *Expert UI* and, of course, the SLPS *Data Access Layer*.

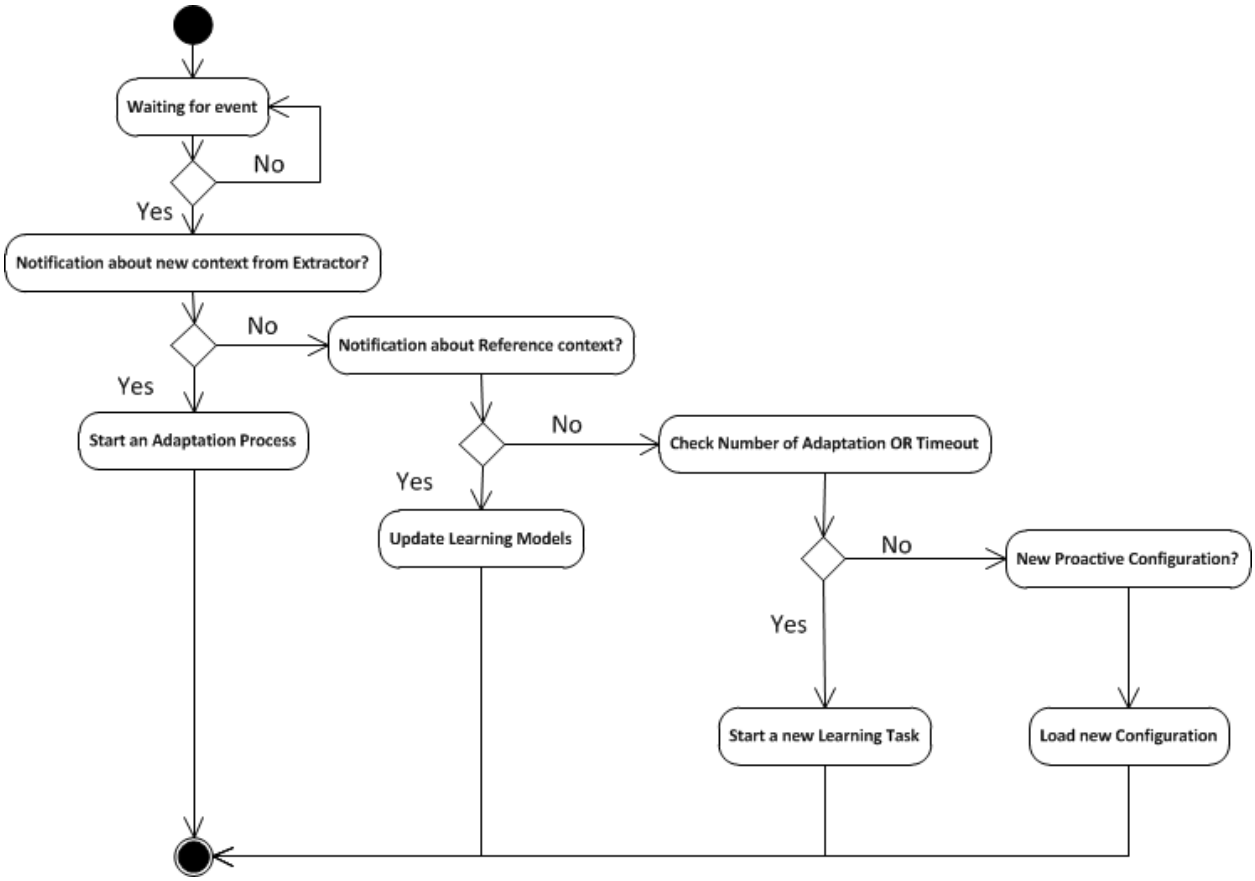


Figure 4.1: SLPS *Adapter* activities during system operation

Since the communication/integration between the different SLPS components is implemented using a *SOAP* compliant web services platform then the SLPS *Adapter* component is interfaced with the other components of the SLPS platform by an *IAdapterService* interface. The *IAdapterService* implements and extends the standard *ISelfLearningService* interface containing a set of web methods enabling the interaction with the SLPS *Adapter* by invoking the required web operations.

The services provided by the SLPS *Adapter* to the SLPS platform are detailed in listing 4.3 and include the following web methods: *config*, *ProactiveConfig*, *informaAboutContextChange*, *informAboutReferenceContext* and *informAboutMonitoredData*.

Listing 4.3: IAdapterService

```
1 @WebService(name = "AdapterService", targetNamespace = "http://selflearning.eu")
2 @SOAPBinding(style = SOAPBinding.Style.DOCUMENT)
3 public interface IAdapterService extends ISelfLearningService {
4
5   @WebMethod(operationName = "config")
6   public void config(
7       @WebParam(name = "application-scenario") ApplicationScenario appScenario,
8       @WebParam(name = "Mode") String Mode,
9       @WebParam(name = "Algorithm") String algorithm)
```

```
10        throws SelfLearningFault;
11
12    @WebMethod(operationName = "Proactive_config")
13    public void ProactiveConfig(
14        @WebParam(name = "application-scenario") ApplicationScenario appScenario)
15        throws SelfLearningFault;
16
17    @WebMethod(operationName = "informAboutReferenceContext")
18    public boolean informAboutReferenceContext(
19        @WebParam(name = "application-scenario") ApplicationScenario appScenario,
20        @WebParam(name = "identifier") String identifier)
21        throws SelfLearningFault;
22
23    @WebMethod(operationName = "informAboutContextChange")
24    public boolean informAboutContextChange(
25        @WebParam(name = "application-scenario") ApplicationScenario appScenario,
26        @WebParam(name = "identifier") String identifier)
27        throws SelfLearningFault;
28  }
```

The *config* web method is used to configure the SLPS *Adapter* component, i.e. to instantiate all its kernel modules that in turn will be used during the adaptation process, as well as, during auxiliary activities such as the updating of learning models of the process and the execution of new learning tasks. All the information necessary for the SLPS *Adapter* component configuration is contained into XML files, as shown in listing 4.4 that details the configuration file for the Bosch Rexroth scenario. For the remaining application scenarios the structure will be exactly the same except for the two parsers, SLPS *Expert UI* and *MonitoringData* components that depend on the particular application scenario.

Listing 4.4: Example of a XML configuration file

```
1  <adapterConfig>
2     <ApplicationScenario>
3          IdleTimeDetection
4     </ApplicationScenario>
5     <RepositoryParser>
6          pt.uninova.selflearning.parsers.RepositoryParserTimeInterval
7     </RepositoryParser>
8     <LearningParser>
9          pt.uninova.selflearning.parsers.LearningParserTimeInterval
10    </LearningParser>
11    <ui>
12         pt.uninova.selflearning.ui.TrayApp
13    </ui>
14    <adapter>
15         pt.uninova.selflearning.adapter.comm.CommunicationAdapter
16    </adapter>
17    <MonitoringDataModel>
18         de.atb.selflearning.monitoring.models.idletime.IdleTimeMonitoringData
19    </MonitoringDataModel>
```

```
20 </adapterConfig>
```

Furthermore, looking for the header of the *config* web method several parameters are passed. The *ApplicationScenario* parameter allows to instantiate the particular implementations of the modules according to the different application scenario in which the SLPS platform is inserted. Moreover, the parameters *Mode* and *Algorithm* are related with configuration issues of the *Learning module* and specify the type of processing to execute and the learning algorithms to use. The *Mode* parameters can assume two possible values, namely *single* and *multi* with the following meaning:

- **single**: means that one learning algorithm and one model are used for make adaptation proposal about the current context encapsulated into the monitoring data structure from the SLPS *Extractor*;
- **multi**: means that the number of learning algorithms and models to use depends on the particular monitoring data structure from the SLPS *Extractor*.

The *Algorithm* parameter can assume one of these values, namely *RuleInduction*, *NeuralNetwork*, *NaiveBayes*, *SupportVectorMachine*, *ID3* or *PolynomialRegression* to define the particular algorithm to use. This parameters is defined according to the nature of the problem in which the SLPS platform is supposed to operate.

The *ProactiveConfig* web method is used to notify the SLPS *Adapter* component that a new configuration for its internal module *AdapterProactiveBehaviour* is available.

The *InformAboutReferenceContext* web method is used to inform the SLPS *Adapter* component that a new reference context is available. The reference context represents a perfectly known state or working condition of the manufacturing production process and is used to populate the learning model of the process bypassing the adaptation process. The *identifier* parameter is used to retrieve the reference monitoring data from the *MonitoringDataRepository*.

Finally, the *InformAboutContextChange* web method triggers an adaptation process whenever a context change is detected by the SLPS *Extractor*. Each adaptation process runs on its own Java thread and a new one is launched for each context change notification sent by the SLPS *Extractor*.

4.2.2.3 Adaptation Process

The adaptation process is triggered every time the SLPS *Extractor* notifies the SLPS *Adapter* about a change in manufacturing production line operative context. When launched, the adaptation process orchestrates a set of tasks and activities to be executed by the internal components of the SLPS *Adapter* with the objective of adapting the manufacturing production system to face the new operative context. The adaptation process has been implemented as a Java thread and is generic enough to cope with different manufacturing production processes. The kernel components of the SLPS *Adapter*, that are invoked

during the execution of an adaptation process, are presented in the following sections focusing on the implementation details.

4.2.2.3.1 Context Change Handler

The *Context Change Handler* component allows to receive the SLPS *Extractor* notification whenever a change of context occurs and to trigger the adaptation process to handle this change. The pseudo-code in algorithm 1 details the activities executed by this component.

```
Require: All Adapter components instantiated
Ensure: New Adaptation process started
initialization;
if notification from Extractor then
    Adaptation process is started for context with identifier contextID;
else
    do nothing;
end
```

Algorithm 1: Context Change Handler activities

4.2.2.3.2 Repository Extractor

The *Repository Extractor* component is responsible for retrieving the context, encapsulated into the monitoring data structure, that has triggered the related adaptation process from the SLPS *Data Access Layer* repositories using a web service based communication. The retrieved monitoring data comprises all the sensory information related with the particular application scenario and is, then, sent to the *Repository Parser* component. The pseudo-code in algorithm 2 details the activities executed during the retrieving process.

```
Require: Monitoring Data Repository Web Service deployed
Ensure: Monitoring Data object
get Monitoring Data with ID = contextID from Data Access Layer;
if Monitoring Data != null then
    return Monitoring Data
else
    return false
end
```

Algorithm 2: Repository Extractor activites

4.2.2.3.3 Repository Parser

After the *Repository Extractor* has retrieved the monitoring data from the SLPS *Data Access Layer*, there is the need to rearrange all the sensory information that it includes using a generic data structure, named *ReasoningInput*, created for this purpose. The *Repository Parser* is the component of the SLPS *Adapter* responsible for receiving the monitoring

data, elaborate all the necessary contextual information in order to create the *ReasoningInput* for the SLPS *Learning Module*. Looking for the implementation aspects, the *Repository Parser* is a Java abstract class that needs to be extended according to the particular application scenario.

The figure 4.2 shows the *RepositoryParser* abstract class and its methods.

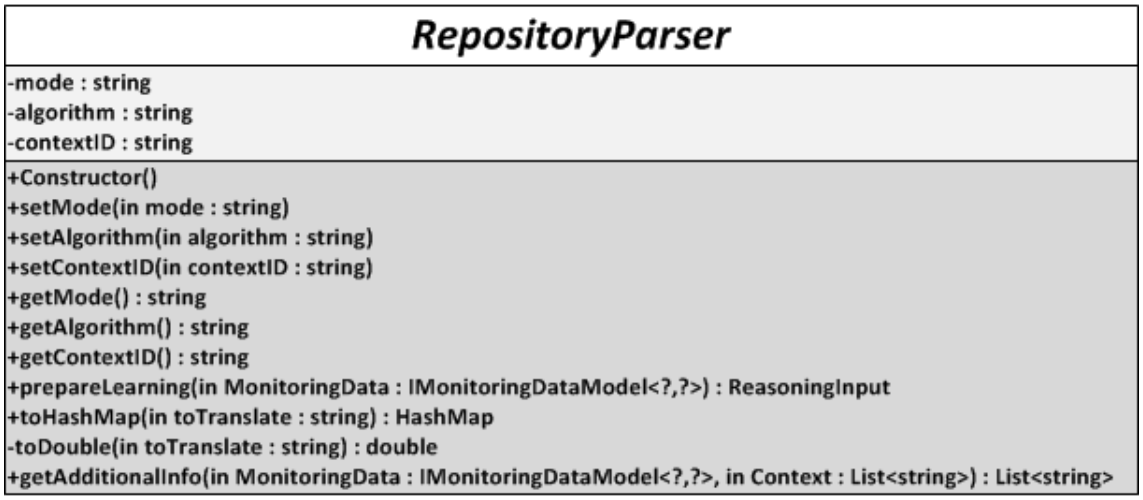


Figure 4.2: *RepositoryParser* class diagram representation

Therefore according to the main objectives of the *Repository Parser*, its fundamental operation is represented by the *prepareLearning* method that "parses" the received monitoring data into the *ReasoningInput* generic data structure. The other methods are auxiliary functions that will be used by the SLPS *Learning Module* to allow the correct retrieving of the information to send to the learning algorithms and will be presented with more details in section 4.2.3.

For each application scenario a suitable *RepositoryParser* class has been implemented to be used during the adaptation process. The loading of the appropriate classes is defined through the *config* web method of the SLPS *Adapter* performed during the system start up.

The *ReasoningInput* is a simple Java class used as wrapper and comprises the following information: *Algorithm*, *Context*, *Mode*, *Description* and *Data* as presented in figure 4.3.

The *Context* attribute contains the identifier of the current context that has triggered the adaptation process. The *Algorithm*, *mode* and *contextData* attributes contain necessary information to pass to the *Learning Module* component in a string format. The first contains a string to define the learning algorithm to instantiate. The second contains a string to inform what kind of reasoning process to execute, i.e. use one algorithm and one learning model (single mode) or instantiate multiple algorithms and learning models (multi mode). The former contains a collection of contextual information constructed from the sensory information contained into the monitoring data, that needs to be analyzed by the learning algorithm. Furthermore, the number of contextual information (the size of the *contextData* list) are used to define the number of learning algorithm to instantiate when

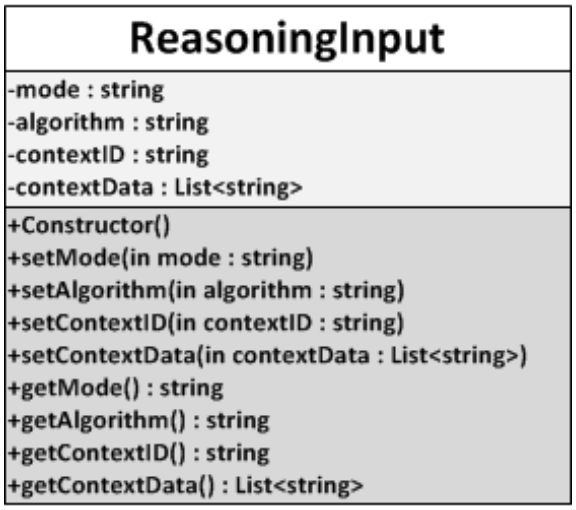


Figure 4.3: *ReasoningInput* class diagram representation

multi mode is selected. The *Algorithm* and *mode* attributes are defined when the SLPS prototype is started and cannot be changed without stopping the running system.

Finally, the activities executed by the *Repository Parser* during an adaptation process are shown by pseudo-code in algorithm 3.

```
Require: Monitoring Data object
Ensure: ReasoningInput object
Initialization;
foreach element e of the Monitoring Data do
    if e needs to be analyzed then
        Insert into contextData collection;
    else
        go to the next element e;
    end
end
return ReasoningInput object
```

Algorithm 3: Repository Parser main activities

4.2.2.3.4 Learning Service

Once the *ReasoningInput* object has been created, wrapping all the data to be analyzed by the SLPS *Learning Module*, it is sent to the SLPS *Learning Module* to start a reasoning process using existing learning models of the process (more details are given in section 4.2.3). The result of this activity, i.e. the adaptation proposal in a raw format calculated by the learning algorithms, is enclosed into the *ReasoningResult* generic data structure and is finally sent to the *Learning Parser*. The *ReasoningResult* is a simple Java class that wraps the SLPS *Learning Module* results and comprises the following information: *contextID*, *result* and *parameterization* as presented in figure 4.4.

The *contextID* attribute contains the identifier of the context to which the adaptation

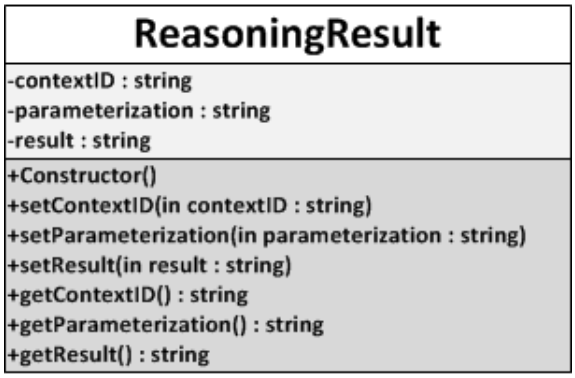


Figure 4.4: *ReasoningResult* class diagram representation

process is related. The *result* attribute is used to store the final result, i.e. the outcome of the learning activity. Finally the *parameterization* attribute contains the set of parameters, i.e. all the necessary sensory information about context gathered from monitoring data, used during the learning activity to determine the result.

4.2.2.3.5 Learning Parser

As stated in 4.2.2.3.4, the result of learning activity executed by the learning algorithm is stored in a generic data structure named *ReasoningResult*. Due to the fact that the information stored is in a raw form, there is a need to "parse" and rearrange it into an appropriate *Adaptation* object. In this context, the *Learning parser* is the SLPS *Adapter* component responsible to rearrange all the information contained in the *ReasoningResult* into a generic form to be shown by the SLPS *Expert UI*. However, it is also used to perform the inverse operation, i.e. receive the *Adaptation* from the *Expert UI* and organize it to be redirected to the SLPS *Learning Module* for updating the learning models of the process. The *Learning parser*, in the same way of the *Repository parser*, is a Java abstract class that needs to be extended according to the particular application scenario, i.e. for each application scenario an implementation of the abstract *Learning Parser* class will exist.

The figure 4.5 shows the *LearningParser* abstract class and its methods.

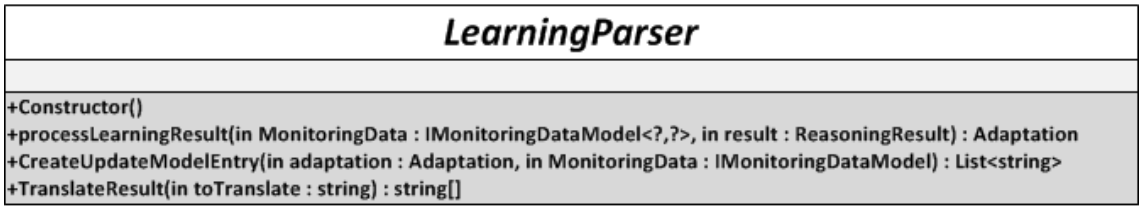


Figure 4.5: *LearningParser* class diagram representation

Two main tasks are performed by the *Learning parser* component during its lifecycle, namely: processing the learning result and creating a new entry for the learning model of the process. The method *processLearningResult* receives the monitoring data and the *ReasoningResult* object to create and *Adaptation* object instance that will be sent to the

SLPS *Expert UI*. The method *CreateUpdateModelEntry* implements the inverse operation, i.e. receives the *Adaptation* object, at this point validated by the expert using the SLPS *Expert UI* commands, and invoke the appropriate learning operation provided by the SLPS *Learning Module* for updating the learning model of the process with new information available.

The *processLearningResult* method returns an *Adaptation* object, representing an instance of the Java class *Adaptation* (see figure 4.6) that comprises the following information: *contextID*, *adaptationID*, *timeStamp*, *result*, *proposal*, *Description* and *textualDescription*.

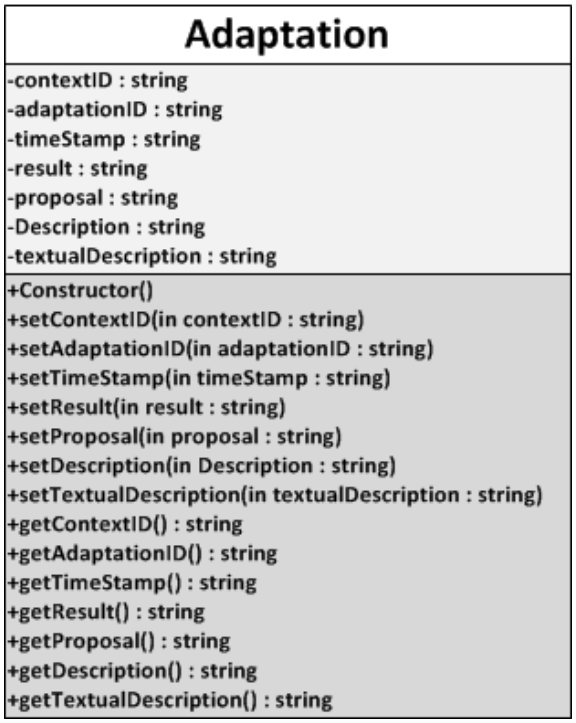


Figure 4.6: *Adaptation* class diagram representation

The *contextID* and the *adaptationID* attributes are the identifier of the current context that has triggered the adaptation process and of the adaptation itself respectively. The *timeStamp* attribute contains information for identifying when the adaptation process has occurred. The *proposal* attribute stores the adaptation proposal or, in other words, the result of the learning activity in a well-defined format, that will be presented to the system expert for validation. The *result* attribute contains the final adaptation result, i.e. the adaptation proposal after the validation activity performed by the system expert. Finally, the *Description* and *textualDescription* attributes are both used to store additional information about the context that has triggered the adaptation process, in particular the former contains the numerical values of the sensory information analyzed by the learning algorithm while the latter contains a human readable description about the context itself.

The *Adaptation* class template is a generic data structure that can be applied to each application scenario, however the *proposal* and *result* attributes, as well as, the *Description*

and the *textualDescription* depend on the particular application scenario and therefore will have a different structure according to this.

The activities and tasks executed by the *Learning Parser* during an adaptation process are shown by the two pseudo-codes in algorithms 4 and 5. The first one details the conversion from *ReasoningResult* to *Adaptation* object, while the second details the conversion from *Adaptation* to *ModelEntries* collection. The *ModelEntries* collection is a List of string containing the new information, retrieved from the result attribute of *Adaptation* object after the validation task performed by the system expert, that will be used by the SLPS *Learning Module* for updating the learning models of the process.

```
Require: ReasoningResult object
Ensure: Adaptation object
Initialization;
foreach element e of the ReasoningResult do
    Insert into Adaptation proposal as string;
end
return Adaptation object
```

Algorithm 4: Learning Parser *ReasoningResult* to *Adaptation* activity

```
Require: Adaptation object
Ensure: ModelEntries as List<string>
Initialization;
foreach element e of the Monitoring Data do
    if e needs to be analyzed then
        Insert into ModelEntries collection;
    else
        go to the next element e;
    end
end
return ModelEntries
```

Algorithm 5: Learning Parser *Adaptation* to model entry collection activity

The conversion from *Adaptation* to *ModelEntries* collection, shown in listings **??**, is skipped whenever the result and the proposal attributes of the *Adaptation* object are the same, meaning that no new knowledge is added by the system expert during the validation task.

4.2.2.3.6 Expert Collaboration UI communication

Whenever an adaptation process is triggered and a new *Adaptation* object is created, containing a proposal for the adaptation of the parameters of the manufacturing process, the SLPS *Adapter* should be able to communicate this new available information to the SLPS *Expert UI* to be displayed to system expert. Furthermore, after system expert has validated the adaptation proposal, the SLPS *Expert UI* should be able to retransmit the *Adaptation* object, that at this point will contain both adaptation proposal and result, back

to the SLPS *Adapter* to be stored and eventually used to improve future adaptation proposals accuracy. In this scenario, the *Expert Collaboration UI communication* module (*UI Comm*) will be responsible for the interaction between the SLPS *Adapter* and SLPS *Expert UI*, providing a communication channel for both transmitting the *Adaptation* object to the user interface and receiving the same *Adaptation* object after the validation task performed by the system expert. Moreover, the *UI Comm* provides also a way to communicate with the *Adapter Proactive* behaviour as exposed in section 4.2.2.4.

The *UI Comm* component has been implemented through a generic Java interface (see figure 4.7) ensuring in this way the Adaptation process abstraction. Moreover, the *UI Comm* interface will be implemented according to the particular application scenario.

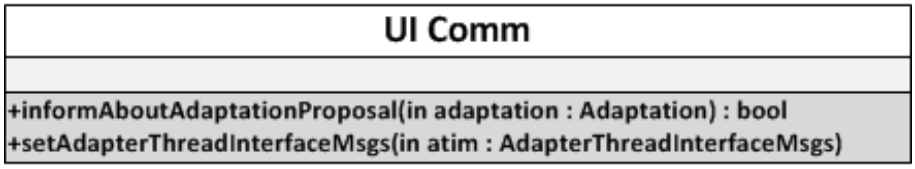


Figure 4.7: *UI Comm* class diagram representation

4.2.2.3.7 Adaptation Distribution

Once the *Adaptation* object, containing the final adaptation result to face the current change of context, is sent back the SLPS platform the adaptation process can be considered concluded. However, several tasks have to be executed before terminating the adaptation process thread, in the order: invoke *Learning Parser* functionalities for updating learning models of the process if new information is available, store the *Adaptation* object into the SLPS *Adaptation Repository* and finally notify the SLPS *Extractor* that an *Adaptation* has been deployed. The *Adaptation Distribution* is the logical component that will perform these tasks. The pseudo-code in algorithm 6 shows the activities executed by the *Adaptation Distribution* component.

Require: Adaptation object
Ensure: Learning Models Updating, Adaptation object stored,Notification to SLPS Extractor
Initialization;
if *Adaptation result != Adaptation proposal* **then**
  call Learning Parser for Learning models updating;
else
  do nothing;
end
Store Adaptation object into AdaptationRepository;
Notify SLPS Extractor;
return *True*

Algorithm 6: Adaptation Distribution activity

4.2.2.4 Adapter Proactive behaviour

The SLPS *Adapter* exhibits a purely reactive behaviour since it waits for notification from the SLPS *Extractor* to start an adaptation process. However, autonomy and proactivity has been also embedded for allowing the SLPS *Adapter* to independently start some activities in certain and suitable point in time. For this purpose, a Java thread named *AdapterProactiveBehaviour*, has been implemented and attached to the SLPS *Adapter* and it is started when the SLPS *Adapter* component is configured. After configured using an appropriate txt configuration file, the *AdapterProactiveBehaviour* will continually wait for the occurrence of several preprogrammed events that in turn will trigger the execution of different tasks according to the type of the event. An *enum* Java type is used to define the different kind of event as shown in listing 4.5

Listing 4.5: Event Type enumeration

```
1 public enum EventType {
2  ADAPTERPROCESS, TIMEOUT, PROACTIVECONFIG, UPDATEFACTOR, UPDATEMODEL, UNKNOWN
3 }
```

The following events has been considered:

- **AdapterProcess**: this event is raised whenever an adaptation process is started, and is used to count the number of adaptation process executed since the last time that a model of the process has been inferred in order to verify if the model is out-of-date.
- **Timeout**: this event is raised whenever the SLPS *Adapter* is waiting for a notification of the SLPS *Extractor* and a predefined time limit is spent and is used to identify suitable point in time to start new learning task.
- **ProactiveConfig**: this event is raised whenever a new configuration for the proactive behaviour is available.
- **UpdateModel**: this event is raised whenever is necessary to update the learning models of the process.
- **Unknown**: is a default event meaning that no event is available.

The occurrence of an event is communicated to the *AdapterProactiveBehaviour* by using a proper interface called *AdapterThreadInterfaceMsgs*. The *AdapterThreadInterfaceMsgs* is a simple Java class providing methods to be used for notifying the *AdapterProactiveBehaviour* during its execution as shown in figure 4.8.

The following methods are available and used during the SLPS *Adapter* lifecycle to communicate the occurrence of a particular event to the *AdapterProactiveBehaviour*: *putEvent*, *putEvent_Config*, *putEvent_Factor* and *putEvent_Update*. The *putEvent* method is used to communicate the starting of a new adaptation process, while the *putEvent_Config* method is used to communicate that a new configuration for the *AdapterProactiveBehaviour* is available. The *putEvent_Factor* method is used to calculate the learning factor. The *putEvent_Update*

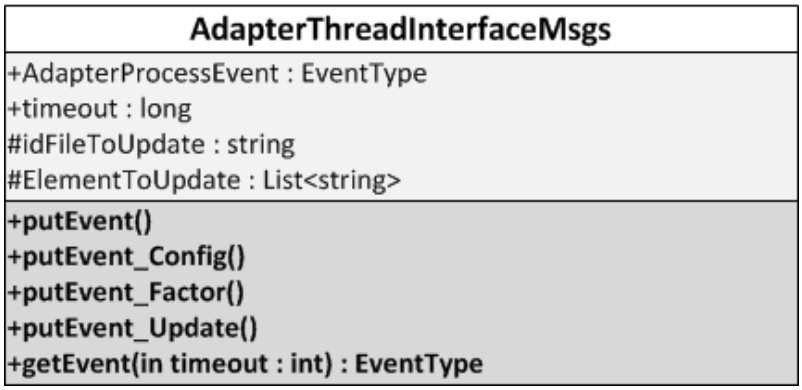


Figure 4.8: *AdapterThreadInterfaceMsgs* class diagram representation

method is used whenever there is the need to change something in the learning models of the process. Moreover, the method *getEvent* is used by the *AdapterProactiveBehaviour*, after being notified, to retrieve the event.

The way the *AdapterProactiveBehaviour* operates strongly depends on how it has been configured. The *AdapterProactiveBehaviour* can be configured, i.e. the level of proactivity of the SLPS *Adapter* is customizable, using a text file which comprises the following set-up parameters:

- **Counter**: is the number of adaptation processes after that the learning model of the process can be considered outdated.
- **Time**: is the maximum time that the SLPS *Adapter* will wait for a notification of the SLPS *Extractor*.
- **Factor**: when the SLPS platform is running and the learning models of the process are dynamically populated, the *Counter* parameter needs to be recalculated along time to reflect the evolution of the models, i.e. when the learning models are empty make sense to update the learning models more frequently than the case in which the learning models have more entries. In this direction the parameter factor is used to recalculate the number of adaptation process after that the learning model of the process can be considered out-dated.

The main objective of the *AdapterProactiveBehaviour* is looking for the occurrence of particular situations identified by determined events to trigger a *Learning Task*. The activities/tasks performed by this component during the SLPS system lifecycle is shown by the pseudo-code in algorithm 7.

4.2.2.4.1 Learning Task

As exposed in 4.2.2.4, the *AdapterProactiveBehaviour* can start *Learning Tasks* whenever certain events occur, namely: a predefined number of adaptation processes have been performed or a timeout occurs while the SLPS *Adapter* is waiting for a notification from

```
Require: Adapter configured
Ensure: Learning Task execution
Initialization;
repeat
    wait for Event occurrence;
    if Event == Adaptation Process then
        Update Counter;
        Reset Timer;
        if Counter > defined counter limit then
            Start a new Learning Task;
        else
            Increment Counter;
        end
    else if Event == Timeout then
        Start a new Learning Task;
    else if Event == Factor then
        Update Defined counter limit;
    else if Event == Proactive Config then
        Load new configuration for Proactive behaviour;
    else if Event == Update model then
        Update Model of the process;
until Adapter is running;
return True
```

Algorithm 7: Adapter Proactive behaviour activities

the SLPS *Extractor*. The *Learning Task* comprises two main operations: start a learning activity allowing the algorithms to update their own models of the process and, after that, start a validation process to evaluate the capabilities of the learning algorithms with the new models. This operations/activities are executed exploiting SLPS *Learning Module* functionalities.

The *Learning Task* implements a Java runnable interface allowing its execution by an independent thread.

4.2.3 Learning Module

4.2.3.1 Behaviour

The SLPS *Learning Module* is a fundamental extension of the SLPS *Adapter* representing its reasoning entity and including all the knowledge about the process in which SLPS platform is inserted. The SLPS *Learning Module* is responsible to draw conclusions about system parameters adjustment considering the actual context of the manufacturing production line, as well as, its internal learning models of the process relying on machine learning techniques. The result of this activity is passed to the SLPS *Adapter* for further processing. Moreover, the SLPS *Learning Module* operations are also invoked, whenever

a *Learning Task* is launched by the *AdapterProactiveBehaviour*, to improve SLPS *Adapter* accuracy of the proposals along time. Although the SLPS *Learning Module* functionalities and features are only exploited by the SLPS *Adapter* during the adaptation process and also during the Proactive behaviour scanning, it remains a multipurpose component that is available in the platform and provides its machine learning services and capabilities in a generic form.

4.2.3.2 Communications

The SLPS *Learning Module* is a passive component inside the overall SLPS architecture and need to be connected to the SLPS *Adapter* to entirely fulfill its objectives and goals, i.e. without the SLPS *Adapter* this component is not used by the other components inside the SLPS architecture. Therefore, the capability of the SLPS solution to reason, adapt and learn during its entire lifecycle strictly depends on the interactions between the SLPS *Learning Module* and the SLPS *Adapter*. For this purpose, the *ILearningService* interface has been implemented, ensuring an entry point to the SLPS *Learning Module*, i.e. provides to the SLPS platform a set of services allowing the exploitation of the SLPS *Learning Module* capabilities.

The services/operations provided by the SLPS *Learning Module* to the SLPS *Adapter* are detailed in listing 4.6 and include the following Java methods: *startRM*, *selectAlgorithm*, *reason*, *learn*, *updateModelEntry*, *updateModel* and *validateModel*.

The communication mechanism between the SLPS *Learning Module* and the SLPS *Adapter* has been implemented using simple Java methods since the SLPS *Learning Module* is attached and used only by the *Adapter*. Furthermore, this decision does not affect/prejudice the concept, the methodology, as well as, the potentiality of the SLPS solution at all.

Listing 4.6: ILearningService

```
1  public interface ILearningService {
2
3      public void startRM();
4
5      public void selectAlgorithm(ReasoningInput inpt);
6
7      public ReasoningResult reason(ReasoningInput inpt);
8
9      public void learn(ApplicationScenario appScenario);
10
11     public void updateModelEntry(ApplicationScenario appScenario,
12                                  List<String> ModelEntry,
13                                  ReasoningInput inpt);
14
15     public boolean updateModel(ApplicationScenario appScenario,
16                                String id,
17                                List<String> ElementToUpdate);
```

```
18
19      public boolean ValidateModel(ApplicationScenario appScenario);
20  }
```

The *startRM* operation is used to initializes all the *RapidMiner* libraries and functionalities that are necessary to employ the necessary learning algorithms. The *selectAlgorithm* operation will instantiate the algorithm(s) to be used during the Adaptation process, i.e., based on the *ReasoningInput* object structure and the information that it wraps, the SLPS *Learning Module* will be able to instantiate the necessary number of algorithms to be used from this point forward. As a matter of fact, when *ReasoningInput* is instantiated, its *contextData* attribute is populated using the data included into the *MonitoringData* that has been retrieved from the SLPS *Data Access Layer*. Therefore, the *contextData* will contain a list of contexts representing the actual/current status of the manufacturing process, the size of this list allows the SLPS *Learning Module* to know exactly how many algorithms instantiate.

The two core operations *learn* and *reason* ensure the training of the algorithms and the calculation of an adaptation proposal. The former will trigger the creation of an explicit model of the process relying on gathered data and selected algorithm. The latter will infer adjustments for process parameters by exploiting the models previously calculated. The quality of the adjustments, i.e. of the proposal calculated by the SLPS *Learning Module*, varies depending on the quality of the learned models. The number of necessary models, used to train the instantiated learning algorithms, as well as, their internal structure strictly depends on the particular application scenario.

The *updateModelEntry* operation will be used for updating the existing learning models of the process with the last Adaptation process result. These new models will be then available for future learning tasks. The *updateModel* operations can be used to delete entries contained inside the learning models of the process. Finally, the *ValidateModel* is used during a *Learning Task* to perform an X-Cross validation for estimating the performance of a predictive model, i.e. how well the model created from the collected data is expected to perform on future as-yet-unseen data.

4.2.3.3 Learning Algorithms

4.2.3.3.1 Definitions and Basic Concepts In the scope of the Self-Learning project the following set of learning algorithms are considered and supported by the SLPS *Learning Module*, namely:

- **Iterative Dichotomiser 3 (ID3) Learner**: The ID3 learner is a basic top-down decision tree algorithm. The procedure used to build the decision tree is based on the concept of entropy. The ID3 basic steps to construct a decision tree can be stated as follows:
  - *Step 1.* Given a collection S of examples calculate the initial value of the entropy using the following expression:

$$Entropy\ (S)\ = \sum_{i=1}^{N} f\ (j) * log_2 f\ (j)$$

Where:

- f: is determined on the basis of the frequency of the occurrence of the value j into the set S.
- n: is the number of different values of the attribute in S.
- *log*: is the binary algorithm.

– *Step 2.* For each attribute in the example set S estimate the information gain that measure the effectiveness of an attribute in classifying the training data (higher is better).

$$G(S, A) = Entropy\ (S) - \sum_{i=1}^{m} f\ (A_i)\ E\ (S_{A_i})$$

Where:

- G: is the gain of the set S after a split over A attribute.
- E: is the entropy of the set S.
- $f\ (A_i)$: is the frequency of the example possessing A_i as value for A in S.
- A_i: is the i^{th} possible value of A.
- S_{A_i}: is a subset of S containing all the examples for which attribute A has value i.

– *Step 3.* Select the attribute that results in the maximum decrease in entropy as the root of the decision tree.

– *Step 4.* Consider the next level of the decision tree and select the attribute that provides the next greatest decrease in entropy.

– *Step 5.* Repeat the steps 1 through 4 for all the others levels of the decision tree.

- **Naïve-Bayes Learner**: is a simple probabilistic algorithm based on applying the Bayes theorem:

$$P\ (A|B) = \frac{P(B|A) * P(A)}{P(B)}$$

Where:

– $P\ (A|B)$: is the conditional probability of A given B.

– $P\ (A)$: is the prior probability of A.

– $P\ (B|A)$: is the conditional probability of B given A.

– $P\ (B)$: is the prior probability of B.

The Naïve-Bayes algorithm looks at each attribute of a given dataset or example set and determines how that attribute, on its own, affects the prediction. The algorithm assumes that the effect of an attribute on the predicted value is independent from the values of others attributes.

- **Support Vector Machine (SVM)**: comprises a set of methods for supervised learning, applicable to both classification and regression problems. The basic idea behind the SVM technique is to generate an optimal separating hyperplane or a set of hyperplanes from a set of training data L. Each example x_i in L has D attributes and belongs to a class y_i. The generated hyperplane or hyperplanes separate the space into tw o or more parts each one containing the examples belonging to the different classes and can be described by the following function:

$$w * x + b = 0$$

where:

  - w: is normal to the hyperplane.
  - $\frac{b}{\|w\|}$: is the perpendicular distance from the hyperplane to the origin.

The algorithm generates the hyperplanes in order to be as far as possible from the closest example of the considered classes. The figure 4.9 shows the hyperplane generated applying the SVM algorithm to separate examples into two classes H_1 and H_2 respectively.

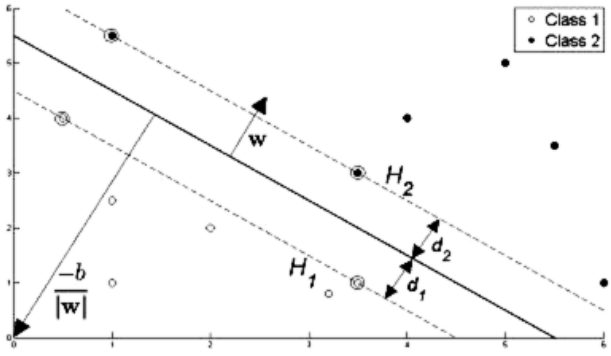


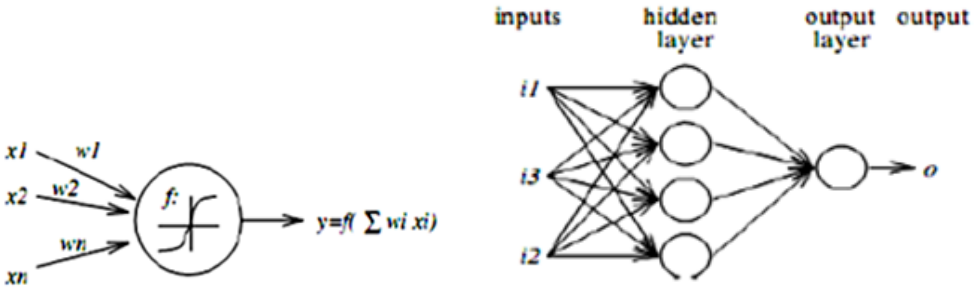
Figure 4.9: SVM classification with two classes

- **Rule Induction**: is a machine learning algorithm that aims to induce a set of complete and consistent rules R from a given example dataset. The rules extracted with Rule Induction algorithm point out hidden regularities in data and define an abstract model representative of the problem from which the data belonging. The rules are expressions of the form:

$$if\,(attribute-1, value-1)\,and \cdots if\,(attribute-n, value-n)\,then\,(decision, value)$$

The set of rules extracted from the example dataset is used mostly to classify new and, above all, unseen cases.

- **Artificial Neural Networks (ANN)**: can be used for supervised learning, in this context the feed-forward multi-layered neural network are most frequently used. The artificial neural network are computational structures consisting of interconnected processing elements (PE) or nodes arranged on a multi-layered hierarchical architecture. In general, a PE computes the weighted sum of its inputs and filters it trough some sigmoid function to obtain the output as shown in figure 4.10(a). The outputs of PEs in one layer serve as inputs to PEs of the next layer as depicted in figure 4.10(b). To obtain the output for a given example set, it is applied to nodes in the lowest layer of network and in each step the outputs of the PEs in the considered layer is computed and passed to the next until the final result is obtained and stored in PEs at output layer.



(a) (b)

Figure 4.10: ANN. (a) Processing Element with sigmoid function. (b) Feed-forward multi-layered neural network.

To use the artificial neural network in classification/regression problems is necessary to train the algorithm using a set of observations in order to adjust its internal weights and move the output in the correct direction (relative to the error between the output of the network and the desired output from the observations). The *back-propagation* algorithm is typically used to train the neural network from a set of observations.

- **Polynomial Regression**: is used to find a functional dependency between the observed training data point. The training data comprises the numeric desired output, called the dependent variable, and the input also called the independent variable. The polynomial regression algorithm aims to find the n^{th} order polynomial function that best fit the complexity of the function inherent in data. Therefore, is possible to model the expected value of y as an n^{th} order polynomial:

$$y = a_0 + a_1 x + a_2 x^2 + a_3 x^3 + \cdots + a_m x^m + \epsilon$$

The learning algorithms presented have been all implemented and embodied in the SLPS *Learning Module* in order to be used during the adaptation process, however their usage and exploiting is limited to the SLPS *Learning Module*, i.e. the learning algorithms have no communication interface to the other component of the SLPS architecture. Moreover, the use of a learning algorithm in respect to another is defined according to the particular application context and nature of the problem (regression or classification) in which SLPS solution is integrated. Classification is the learning of a function that aims to map the data inside a set of classes in the best way possible and is used to identify group membership. On the contrary, regression focuses on the relationship between dependent variable (value to predict) and one or more independent variables and involves estimating and/or predicting the future value of the dependent variable. Furthermore, once the learning algorithm has been chosen and the SLPS platform is started, it will not possible to change algorithm during system runtime. Finally the number of instantiated algorithms is internally managed by the SLPS *Learning Module* (as exposed in section 4.2.3.2) and depends on the structure of the *ReasoningInput*.

4.2.3.3.2 Implementation

Looking for the implementation aspects, all the learning algorithms have been developed using the *RapidMiner* 4.6 API that provides operators for machine learning and data mining application.

The learning algorithms inside the SLPS platform implement a general *IMiner* interface containing all the methods that the implementing algorithm will provide. The standard interface for SLPS algorithm, shown in listing 4.7, provides three basic methods, namely *learn*, *predict* and *ValidateModel*.

Listing 4.7: IMiner

```
1 public interface IMiner {
2
3     public void learn(String dataToLearn);
4
5     public double predict(double[] context);
6
7     public ModelData validateModel();
8
9 }
```

These methods are then implemented accordingly to the type of learning algorithm. The *learn* method is used by the SLPS *Learning Module* to trigger a learning model update: the algorithm will receive all the knowledge about the process, retrieved from the so called *data-to-learn* txt file, in a string format. This information will be used to infer a new up-to-date learning model of the process that will be used in the next adaptation process. The *data-to-learn* text file, storing all the knowledge about the considered manufacturing process, is associated to each learning algorithm and is dynamically filled, i.e.

during an adaptation process if the adaptation proposal is different from the adaptation result a new entry is stored in file. The *predict* method is used by the SLPS *Learning Module* during an adaptation process to calculate a proposal for the current context stored in the *context* array parameter. The double array will contain all the considered parameters for the particular application scenario and is retrieved from the *ReasoningInput* object. However, the information about context contained in the *ReasoningInput* is in a string format, accordingly built by the *Repository Parser* component, and needs to be preprocessed by the SLPLS *Learning Module* using some auxiliary functions provided by the *Repository Parser* in order to have an array of double (see figure 4.11). The result of the execution of the *predict* operation is sent back to the SLPS *Learning Module* which in turn will instantiate the *Reasoning Output* object.

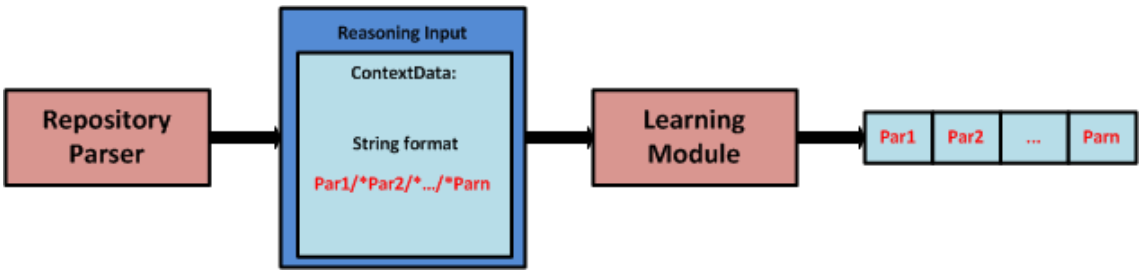


Figure 4.11: Data preprocessing: from String to double array

Finally, the *ValidateModel* method is used by the SLPS *Learning Module* during a *Learning Task* started by the *AdapterProactiveBehaviour*. The *ValidateModel* operation implements a cross (or X-Cross) validation for assessing the capabilities of the model, created by a learning algorithm form the data stored in the text file, to generalize to an independent data set. Even in this case, the Cross validation is performed with the operators provided by the *RapidMiner* 4.6 API and involves three steps:

1. partitioning a sample of data S in S_i subsets;

2. performing the learn on one subset (training set);

3. and validating it on the other subsets (testing set).

The result of the cross validation and the statistics related with all the parameters used to build the learning model are stored into a proper repository named *ModelRepository* and are accessible from the SLPS *Expert UI*.

As stated in section 4.2.1.2, the *ModelRepository* implements the *IRepository* interface to ensure the creation, access and query of the database. In the SLPS platform exist as many *ModelRepositories* as the number of business cases, i.e. the application scenarios grouped in the same business cases share the same repository for storing the cross validation result of learning models.

4.2.3.4 Models Manager

4.2.3.4.1 Behaviour

The *Models Manager* is the component inside the SLPS *Learning Module* responsible for managing the *data-to-learn* text files containing all the necessary data to train the learning algorithms for each application scenario. In the context of SLPS solution, supervised learning technique is used meaning that each learning algorithm infers a function from labeled training data and applies this function to new unseen data/contexts.

The training data is composed by a set of training examples, each one consisting of an input object (the set of parameters used to infer the function) and a desired output (the variable to predict) representing reference/optimum condition of the manufacturing production line. Whenever a learning model update is initiated by the SLPS *Learning Module* or a *Learning task* is started by the *Adapter Proactive Behaviour*, the *Model Manager* will be responsible to extract the all the information contained into the *data-to-learn* text file and sent it to each algorithm as a string. Furthermore, the *Models Manager* is also used by the SLPS *Learning Module* during an adaptation process and, in particular, after the validation of the adaptation proposal performed by the system expert. In this situation, the *Adaptation* object is sent back to the SLPS platform and a new control is performed on it aiming to verify if the adaptation proposal is different from the adaptation result and if it is, the *Models Manager* is used by the *Learning Module* to store this new entry into the related *data-to-learn* txt file in order to be used for future adaptations.

The main activities executed by the *Models Manager* during the SLPS lifecycle are shown in algorithms 8 and 9.

```
Require: filepath destination
Ensure: Data-to-Learn as String for learning algorithm
Initialization;
if file not found then
    return null
else
    open file;
    while !file.eof() do
        Extract Data-to-Learn;
    end
end
return Data-To-Learn
```

Algorithm 8: Models Manager activities during the learning process

4.2.3.4.2 Communications

Considering the general behaviour of the *Model Manager* component inside the SLPS architecture is evident that it is only accessed by the SLPS *Learning Module* during the adaptation process or whenever a new *Learning Task* is started by the *Adapter Proactive*

Require: new Entry for Data-to-Learn file
Ensure: Data-to-learn file updated
Initialization;
if *file not found* **then**
  return *False*
else
  open file;
  write new Entry into Data-To-Learn file
end
return *True*

Algorithm 9: Models Manager activities during the learning Task

Behaviour. The *Models Manager* has been implemented has a simple Java class that implements the interface *IModelsManager* containing the main operations that the *Models Manger* provides to the SLPS *Learning Module*, as shown in listing 4.8. Therefore, the *Models Manager* implement a simple callable unit and need to be instantiated by the SLPS *Learning Module* to be accessible.

Listing 4.8: IModelsManager

```
1  public interface IModelsManager {
2
3    public String getModel(String filepath, ApplicationScenario as);
4
5    public boolean updateModel(String filepath, String FinalResult);
6
7  }
```

The *getModel* method is used to retrieve all the information stored into the *data-to-learn* text file. This information is processed inside the method in order to return a string in a well-defined format to be understood by the learning algorithm. The *updateModel* is used whenever new information about the process is needed to be stored, allowing to populate dynamically the *data-to-learn* text files. This information will be available for future adaptation processes, i.e. when the *getModel* operation will be performed again, the retrieved data from the *data-to-learn* text file will include this new information.

4.2.4 Adaptation Repository

4.2.4.0.3 Behaviour

The final result of each adaptation process started by the SLPS *Adapter* or, in other words the *Adaptation* object instance validated by the user expert, is stored into a dedicated repository called *Adaptation Repository*. This way, it offers to the overall system, constituted by the SLPS platform plus all the software applications inside the manufacturing enterprise which may need, a fundamental collection of data to be taken into account to support lifecycle evolution and system crescent learning performance. As stated in section 4.2.1.2, the *Adaptation Repository* implements the *IRepository* interface to ensure

the creation, access and query of the database. Finally, in the SLPS platform exist as many *Adaptation Repositories* as the number of application scenarios.

4.2.4.0.4 Communications

The *Adaptation Repository* is accessible by all the components inside and outside the SLPS platform through a web service interface containing the set of operations to allow the storing of new *Adaptation* instances and/or the retrieving previous ones. The existing available operations are presented in the listing 4.9 and include the following web method, namely *getAdaptation*, *getLastAdaptationByCount*, *getLastAdaptationByTimeFrame*, *storeAdfaptation*.

Listing 4.9: IAdapterRepositoryService

```
@WebService(name = "AdaptationRepositoryService",
            targetNamespace = "http://selflearning.eu")
@SOAPBinding(style = SOAPBinding.Style.DOCUMENT)
public interface IAdaptationRepositoryService extends ISelfLearningService {

  @WebMethod(operationName = "getAdaptation")
  public Adaptation getAdaptation(
      @WebParam(name = "application-scenario") ApplicationScenario appScenario,
      @WebParam(name = "Adaptation-ID") String adaptationID);

  @WebMethod(operationName = "getLastAdaptationsByCount")
  public List<Adaptation> getLastAdaptations(
      @WebParam(name = "application-scenario") ApplicationScenario appScenario,
      @WebParam(name = "count") int count);

  @WebMethod(operationName = "getLastAdaptationsByTimeFrame")
  public List<Adaptation> getLastAdaptations(
      @WebParam(name = "application-scenario") ApplicationScenario appScenario,
      @WebParam(name = "time-frame") TimeFrame timeFrame);

  @WebMethod(operationName = "storeAdaptation")
  public boolean storeAdaptation(
      @WebParam(name = "application-scenario") ApplicationScenario appScenario,
      @WebParam(name = "adaptation") Adaptation adaptation);

}
```

The *getAdaptation* method is used to retrieve the *Adaptation* instance with the identifier *adaptationID*. The *getLastAdaptationsByCount* method is used to retrieve a collection containing *count Adaptation* instances. The *getLastAdaptationByTimeFrame* methods is used to retrieve a collection containing all the *Adaptation* instances performed by the SLPS solution in the specified *TimeFrame*. The *storeAdaptation* method is used to store a new *Adaptation* instance into the repository.

4.2.5 Functionalities covered by the SLPS prototype: application scenarios presentation

The SLPS prototype has been designed and developed with the participation and support of three industrial partners, ensuring a huge amount of technical knowledge about different kind of manufacturing production processes and useful to point traditional monitoring and control solutions weakness to be tackled and gaps to be filled while improving the feasibility of the prototype.

The efforts of both, academic and industrial world, led to a final SLPS prototype to be validated and assessed in different application scenarios at these industrial partners. Therefore, in order to assure that the proposed solution and infrastructure, as well as, the methodology are generic enough to be applicable at different levels of control and monitoring integration and in different real industrial environments, several application scenarios organized in three business cases have been selected addressing, as above stated, different integration aspects, different manufacturing equipment types and production systems and processes comprising discrete manufacturing industry, specifically machine tools, automotive and aerospace industry. An overview of the BCs is provided in table 4.2.

Overview of considered Business Cases			
Partner	**Core Business**	**Demonstrator**	**Technical issues addressed**
Bosch Rexroth	Control, automation and drive systems	Control Systems of Machine tools	Self-Learning condition based maintenance and energy consumption, adaptation of control strategies to integrate maintenance and energy consumption activities
DESMA	Machines and automation systems for shoe industry	Control systems of machines/automation systems under development	Self-Learning intelligent monitoring approach and context based adaptation of manufacturing process parameters
FASTEMS	Highly customized FMS systems	FMS experimental cell	Self-Learning scheduling and dispatching in Flexible Manufacturing Systems (FMS) for automotive industry

Table 4.2: Overview of considered Business Cases

4.2.5.1 Business Case 1 at Bosch-Rexroth (BR): Self-Learning optimization of secondary processes in CNC machine tools

4.2.5.1.1 Application Scenario: Energy Management

The continuous pursuit of productivity and particularly of machine availability has

led to an increase of the total energy consumption in production plants. Although production machines have become more efficient in terms of accuracy, cycle time and flexibility, there are yet some deficits, such as the efficient handling in respect with resources [Dornfeld, 2010, Gutowski et al., 2006]. Machine tools are responsible for a significant part of energy consumption in manufacturing plants. These machines are extremely complex systems characterized by several functional units to enable a great flexibility in terms of models and designs. A way to improve their energy efficiency can be the increasing of the efficiency factor of the their internal components, however this approach alone is not sufficient reinforcing the idea that a new and more integrated approach for optimizing the management of the resources is peremptory. In this direction, the entire lifecycle of the machine assumes a fundamental importance to define the utilization degree of the whole machine [Schmitt et al., 2011] ensuring the detection of energetic disadvantageous working points and, above all, the identification of unavoidable idle time periods. As a matter of fact, in machine tools with low utilization degree, the machine usually remains in an active state also during non-operating periods due to the availability requirement to avoid production delays. The traditional approach to improve the machine utilization degree is represented by the so called *time-out approach* consisting in defining a *time-out* for each machine subsystem used to shut off the auxiliary services during the idle times. The auxiliary services, i.e. the machine subsystems, are automatically turned on if a new production order is received. This approach improves the machine utilization degree, however the reactive nature of this behaviour implies an extended execution time due to the wake-up delay of the machine subsystems.

The figure 4.12 shows the *time-out approach*.

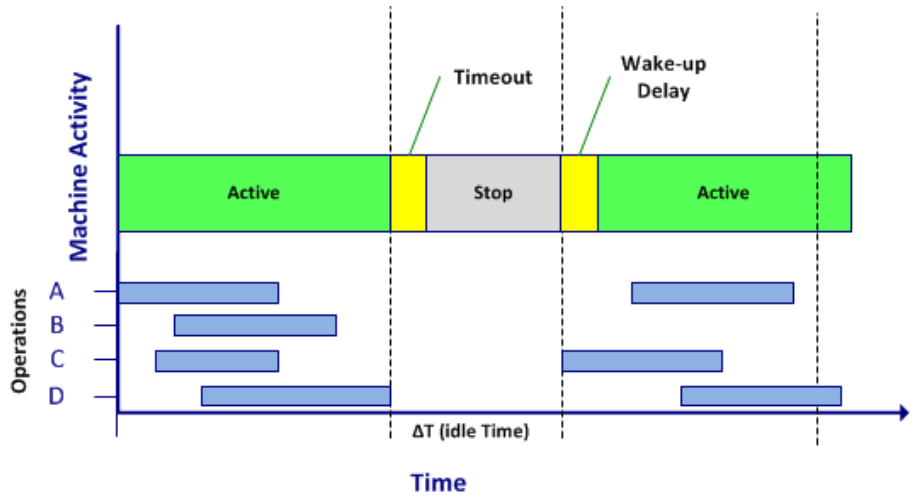


Figure 4.12: Machine Utilization Degree: *time-out approach* [Self-Learning, 2010c]

In this context, the *Self-Learning approach* arises as an alternative to the *time-out approach* for reducing machine energy consumption and improving the machine utilization degree. The *Self-Learning approach* makes use of the statistical data generated from the

continuous observation of the production context, i.e. machine states/operations monitoring, to find machine utilization patterns. These patterns makes it possible to predict the future machine utilization and shut off directly the machine subsystems after the advent of an idle time without waiting for the time-out. Furthermore, if the characteristic wake-up delay is known, it is possible to turn on the machine subsystems avoiding the extend time execution problem. Compared with the traditional *time-out approach*, this solution ensures a deeper interaction between production planning and control layer and process control layer improving the information integration. The figure 4.13 shows the *Self-Learning approach*.

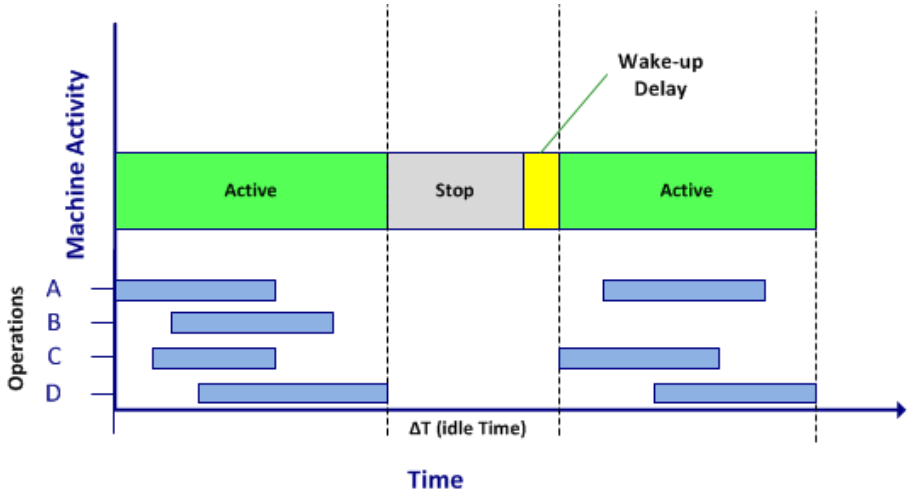


Figure 4.13: Machine Utilization Degree: *Self-Learning approach* [Self-Learning, 2010c]

4.2.5.1.2 Running the SLPS prototype

The implemented SLPS prototype for the Bosch Rexroth application scenario focuses on two main tasks, namely extraction of context information about machine idle time statistical data from the Bosch system and the adaptation of the machine behaviour to the future incoming production activities. The former task is performed by the SLPS *Extractor* component that is responsible to collect machine idle times from machine statistical data (see figure 4.14) and encapsulate it in a standardized meta-model to be sent to the SLPS *Adapter* component.

The latter task is performed by the SLPS *Adapter* that is responsible to explore the received data, looking for suitable idle times, for planning machine energy management activities and communicate the result to the machine system expert as an adaptation proposal. Adaptation suggestions will depend both on temporal idle times dimension and on the entire system lifecycle, i.e. taking into account the different energy tasks executed in the past. As stated before, in the context of this dissertation only the SLPS *Adapter* activities and functionalities are detailed. When starting the SLPS prototype the screen in figure 4.15 will be presented to the user:

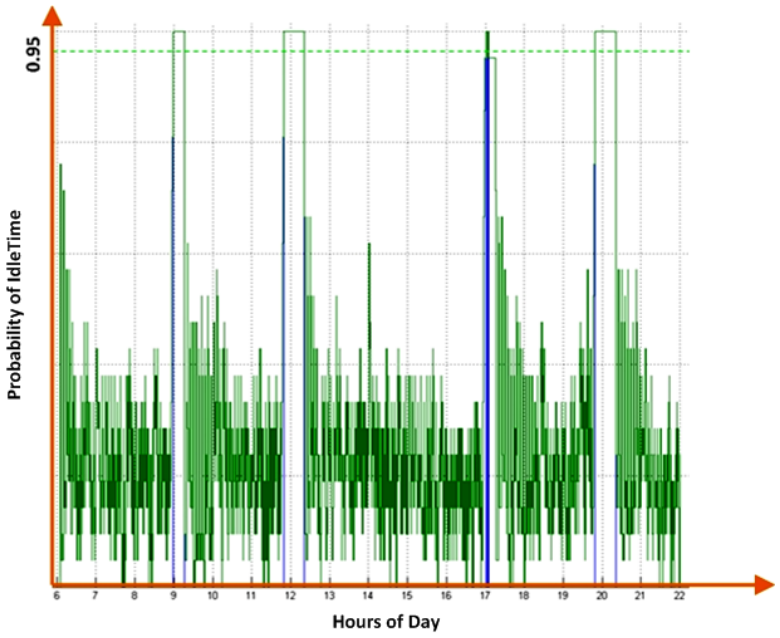


Figure 4.14: Example of detected machine idle times statistical data

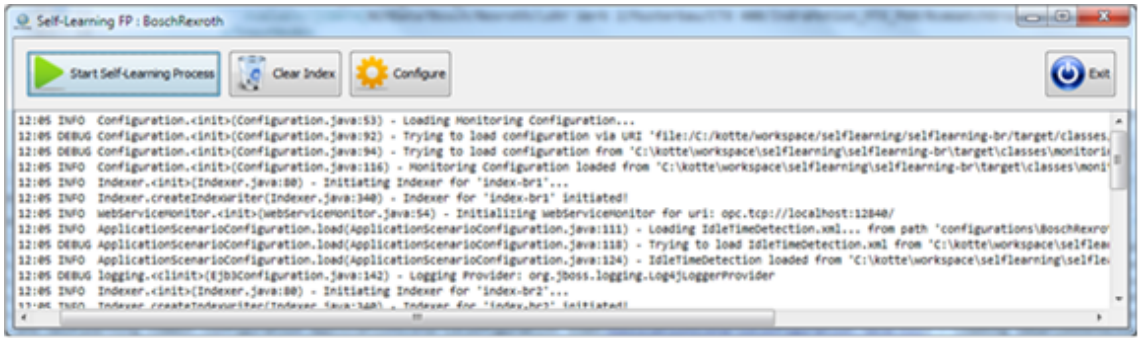


Figure 4.15: SLPS prototype Starter UI for Bosch Rexroth business case

After started the SLPS prototype using the *Start Self-Learning Process* button, the notification shown in figure 4.16 appears into the taskbar notification area.

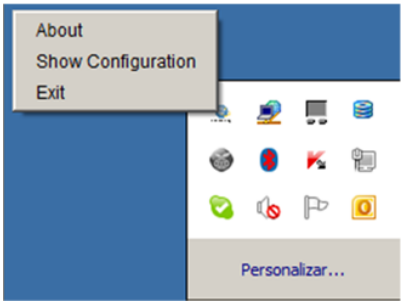


Figure 4.16: SLPS prototype taskbar notification

Clicking on the *Show Configuration* option the configuration UI (see figure 4.17) appears and Self-Learning process is activated.

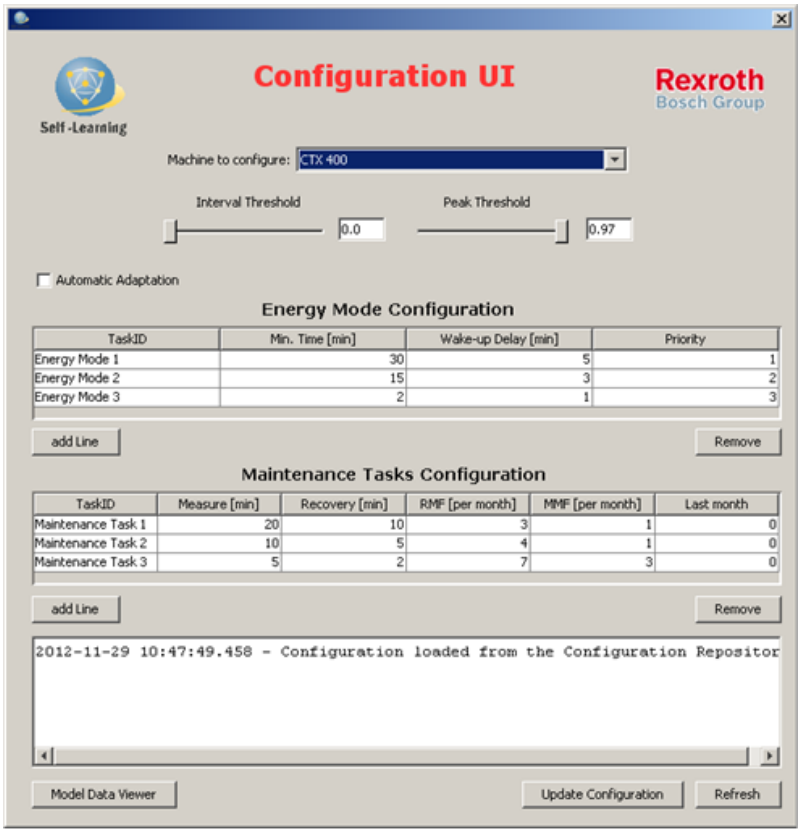


Figure 4.17: SLPS prototype Configuration UI

The configuration UI is always running during the SLPS solution lifecycle and comprises necessary information for identifying relevant idle time windows, as well as, for configuring the energy saving modes and maintenance tasks for the selected machine (*Machine to configure parameter*).

The SLPS prototype has been prepared for both schedule energy saving mode tasks and maintenance tasks, however in the context of this dissertation only the energy saving mode tasks have been considered and tested with the Bosch Rexroth. The *Interval Threshold* and *Peak Threshold* are both used by the SLPS *Extractor* to configure the relevant idle time windows identification task.

The tables *Energy Mode Configuration* and *Maintenance Task Configuration* are initially populated with the information stored into the XML configuration file. However, new energy saving modes and maintenance tasks can be added and/or removed using the *addLine* and *Remove* button respectively. Furthermore, it is possible to modify the existent tasks directly clicking on each element of the table. Moreover, whenever new energy saving mode and maintenance task have been added/removed or modified in the respective tables the *Update Configuration* button can be used to store the new configuration for the machine *Machine to Configure* into the XML configuration file. If an energy saving mode and/or a maintenance task is removed, *Update Configuration* button triggers a communication with the Adapter Proactive behaviour component to update the *data-to-learn* text

file, i.e. open the text file for removing all the entries which result is the eliminated energy saving mode or maintenance task. The *Refresh* button is normally used to retrieve the last saved configuration from the XML configuration file and populate the tables replacing each value with the retrieved ones.

Finally, the *Model Data Viewer* button is used to get the result of the *X-Cross* validation performed automatically by the SLPS *Adapter* during its lifecycle. When clicked, the form shown in figure 4.18 will appear. This form allows the selection of the machine or in other words the selection of the models associated to that machine, and the number of models to retrieve from the SLPS *Adapter* model repository.

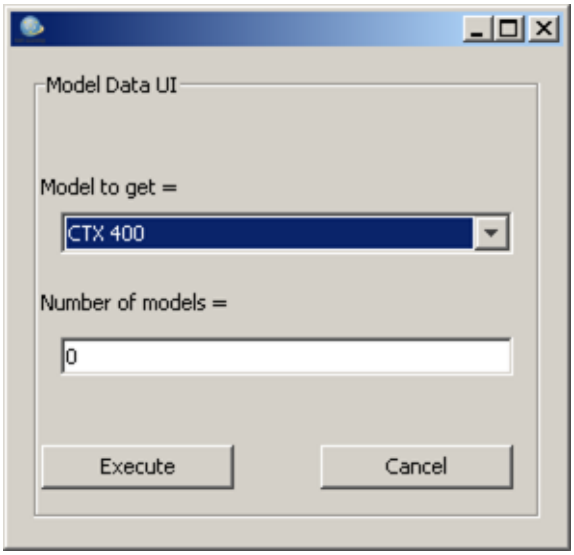


Figure 4.18: SLPS prototype Model Query UI

When the button Execute is clicked the graphs shown in figure 4.19 and figure 4.20 will appear.

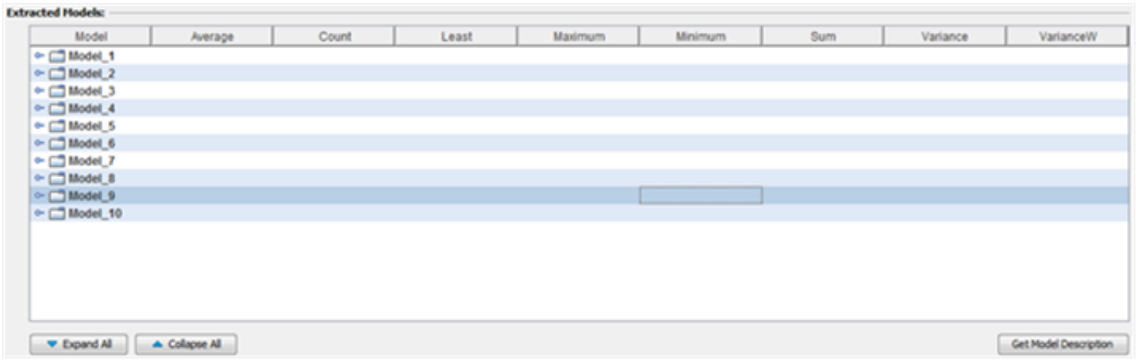


Figure 4.19: SLPS prototype Model Viewer

The *Model Viewer* form is responsible for showing, for each model retrieved form the SLPS *Adapter* repository model, all the information generated during the *X-Cross* validation in a tabular form.

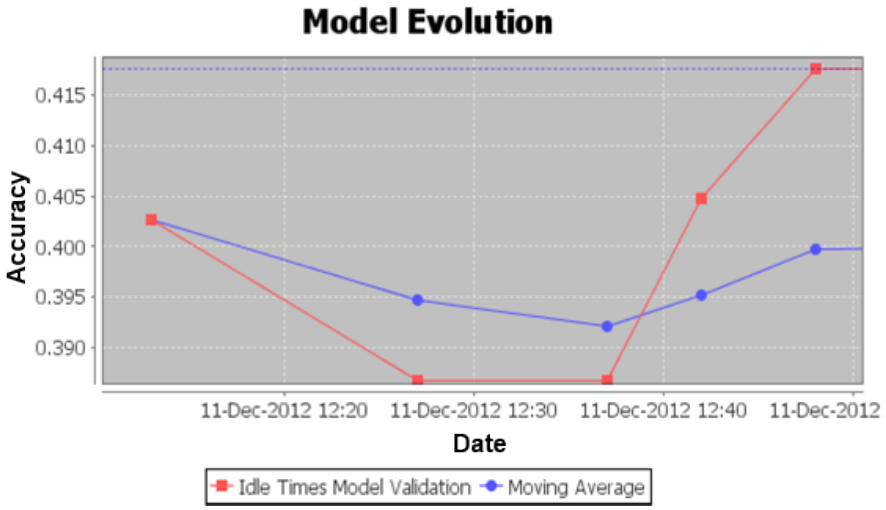


Figure 4.20: SLPS prototype Model Evolution Viewer

The *Model Evolution Viewer* form is responsible for showing graphically the evolution of the accuracy/relative error of the model along time. Furthermore, also the moving average is plotted to have an idea on the real trend of the accuracy/relative error. The graph will plot the accuracy of the model whenever a classification algorithm is used, while the relative error will be plot if a regression algorithm is used. The *Model Data Viewer* functionality can be used by the system expert to analyze indirectly the SLPS *Adapter* performance, from the evolution of the learning models of the process along time, in order to have an idea about the quality of its proposals.

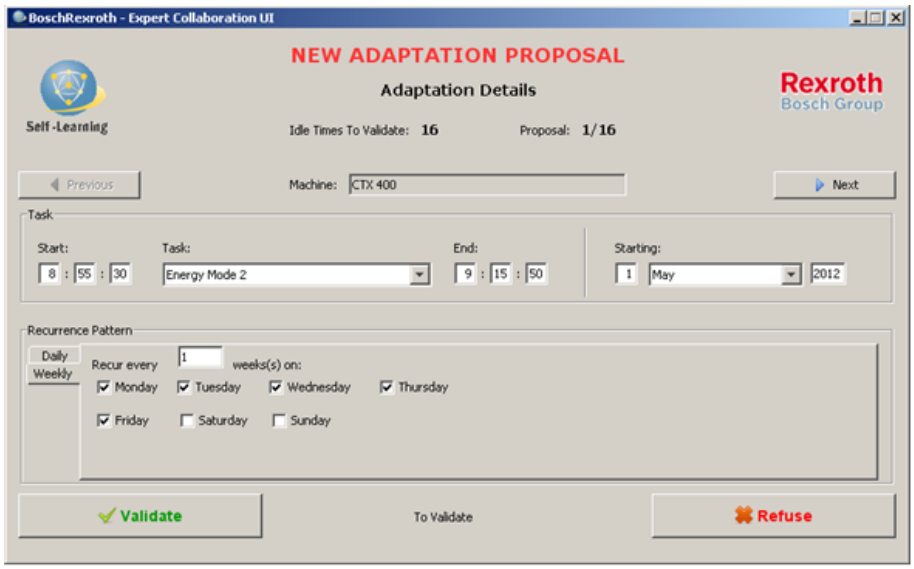


Figure 4.21: SLPS prototype Expert Collaboration UI for Bosch Rexroth business case

The SLPS *Expert UI* (see figure 4.21) and a notification pop-up (see figure 4.22) are both shown whenever a new adaptation is calculated by the SLPS *Adapter* for a machine. All the details about the adaptation proposal are shown in the UI in order to help the

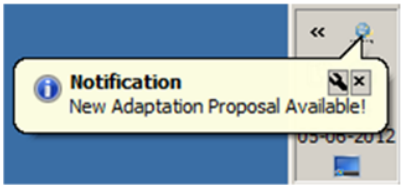


Figure 4.22: SLPS prototype taskbar adaptation notification

system expert during the validation activity. Since the SLPS *Adapter* creates a proposal for each identified idle time then the number of the total proposals as well as the number of validated proposals is shown. For each proposal, the SLPS *Expert UI* shows the starting and ending time, the start date and the proposed task (in the example Energy Mode 2). Furthermore information about the recurrence pattern of the related idle time is shown and could be modified by the system expert if necessary. The *Validate* button is used to both validate the adaptation proposal as it is or eventually modified by the system expert, in other words, the *Validate* button permits to accept all the information shown into the SLPS *Expert UI*. In this case, the idle time together with the selected task will be sent to the machine (in the example machine *CTX-400*). On the contrary, the *Refuse* button is used to refuse the adaptation proposal and the identified idle time will not be considered anymore. Finally, the interaction between SLPS solution and system expert, enabled by the SLPS *Expert UI*, is always needed if the check box *Automatic Mode* (in *Configuration UI*) is not selected. On the contrary, if the check box *Automatic Mode* is selected then no interaction with system expert is expected and all the adaptation proposals will be sent to the Bosch Rexroth system without passing for the system expert validation inspection.

4.2.5.1.3 Learning Algorithm

In this scenario, machines idle time intervals are identified and, based on their duration the SLPS solution is able to take a decision about what to do during these intervals, i.e. which energy saving mode to select according to the particular idle time and the entire system lifecycle. The problem can be modelled as a set of roles that can be fired according to the duration of each interval and the day in which it happens, in this context the *Rule Induction* algorithm represents the best choice. Still, others algorithms can be successfully applied to this scenario such as *Naïve-Bayes* and *ID3* learners.

4.2.5.1.4 Experimental Results

The early experimental results focused on the verification of SLPS solution reliability, feasibility and robustness. During experimentations a set of off-line manufacturing data, extracted from CNC machines during production activity, has been provided by Bosch-Rexroth. This data included machines idle time intervals and related time stamp. The objective was to feed the SLPS solution with the provided data for testing the capability of the system to find patterns in idle times and schedule machine energy mode tasks.

The selected energy mode tasks were then communicated to the existent shop floor machines using an OPC-UA connection. The SLPS solution has been tested for several hours along several days showing good levels of reliability, feasibility and robustness, however further tests are needed in a real manufacturing environment and assisted by a system expert to quantify the real energy saving along the considered time period. At this stage, the SLPS solution experiments have proved to fulfill reliability, feasibility and robustness requirements in a complex industrial setting enabling the integration of the SLPS solution into a real industrial environment. The SLPS solution has been then delivered to the Bosch-Rexroth experts for further analysis and validations. The final report from Bosch-Rexroth regarding energy consumption information before and after the application of SLPS solution shows quite satisfying results. The SLPS prototype has been tested on data gathered from machines in BR-Plant *Lohr Werk 2* (see figure 4.23).

Figure 4.23: Demonstrator used for testing and valdationg the SLPS prototype

An example of extracted data from the machine is shown in figure 4.24.

Finally, the results about the real energy saving together with the loss of machine availability obtained by using the SLPS provided prototype are shown in figure 4.25.

The figure clearly shows that the application of the SLPS prototype results in a real improvement of the energy saving for machine tools. However, as for the machine availability along time the figure shows the presence of an initial transient phase where the energy saving improves while the machine availability decreases due to the learning model of the machine that initially has not enough entries to correctly predict machine behaviour. After this transient phase, the system finally stabilizes, i.e. the SLPS *Adapter* learns with the system expert decisions along time and populates the learning model of the machine with new entries enhancing its capability to generalize. As a result, the loss of availability along time goes to the final zero value.

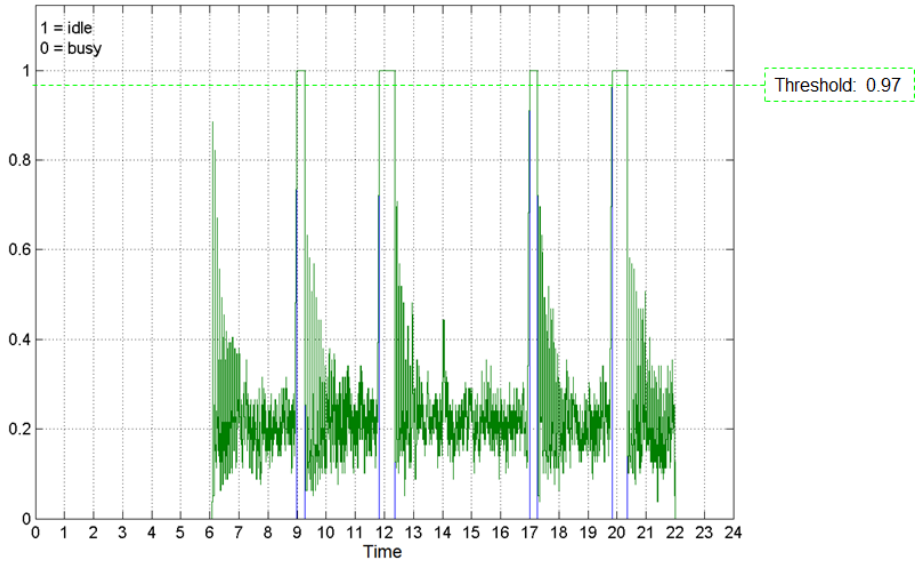


Figure 4.24: idle times statistical data extracted from the machine in one day

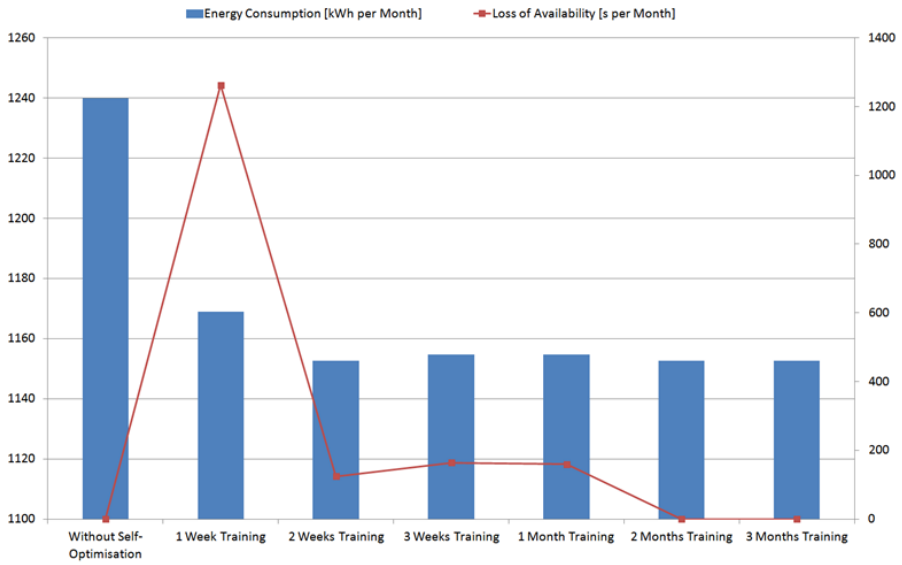


Figure 4.25: Energy saving result and machine availability by using the SLPS prototype

4.2.5.2 Business Case 2 at Desma: Self-Learning intelligent monitoring and adaptation of machines parameters for shoe industry

Today's manufacturing processes are caught between the growing needs for quality, high process safety, efficiency in manufacturing process, reduced time-to-market and higher productivity. In order to meet these demands, more and more manufacturing companies are betting on the application of intelligent monitoring and control solution to reduce maintenance problems, production line downtimes and reduction of production line operational costs with a more efficient management of the manufacturing resources. In this scenario, Desma is intended to face these challenges enhancing its monitoring and control solutions by using novel technologies and especially self-adaptive and context

awareness controls for machines in the shoe productions.

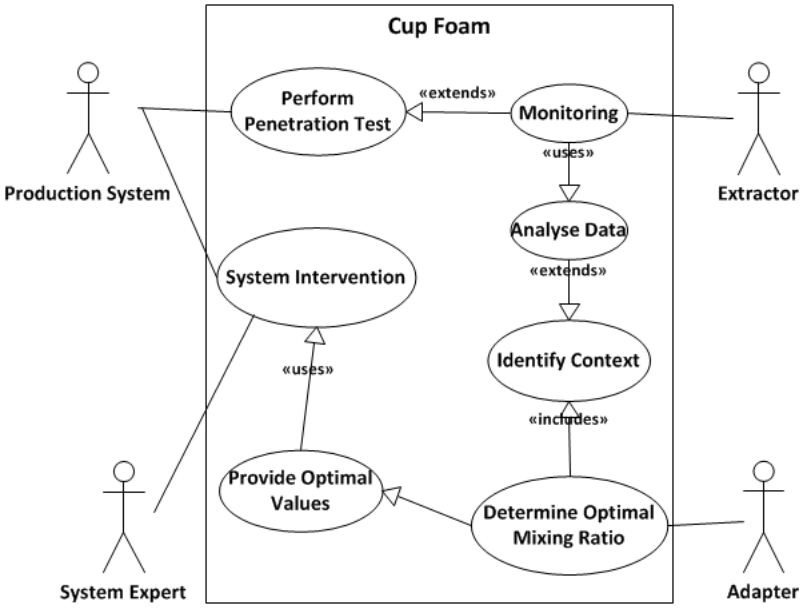
4.2.5.2.1 Application Scenario 1: Cup Foam

Before the starting of shoe production is peremptory to calculate the optimum mixture ratio between the Polyurethen (A) and Polyisocyanat (B) components used to produce the Polyurethan (PUR) component used during the production. For testing the optimum mixture ratio of components A and B of PUR the "cup foam" process is used consisting of a cup and a shaft/pole used to measure the penetration depth. Due to environmental conditions, wear of machine components or fresh material the determination of the optimum mixture ration needs to be carried out regularly. Only the optimum mix ratio makes a good quality product. Today the determination of the optimum mixture ratio is done using a penetration machine. The penetration machine can make clear statements about the foam quality. It allows a weighted pole that slides along a vertical guide, to penetrate into the foam. This penetration is done after a specified time in the rising of the foam of a freshly poured cup. The penetration depth can be read on a scale. Both by under cross linking - as well as over cross linking - the pole penetrate deeper into the foam, than when having the optimum mixture ratio. The whole process described above is done manually by the operator of the machine. This implies that the whole "cup foam" process is fault prone. Therefore DESMA intends to automate this measurement process and integrate it into the production process, to increase measurement correctness, optimize mixing ratio and thereby increase the overall production quality. Therefore the key issue in this application scenario is the automatic testing of the mixing ratio to assure a continuously high production quality.

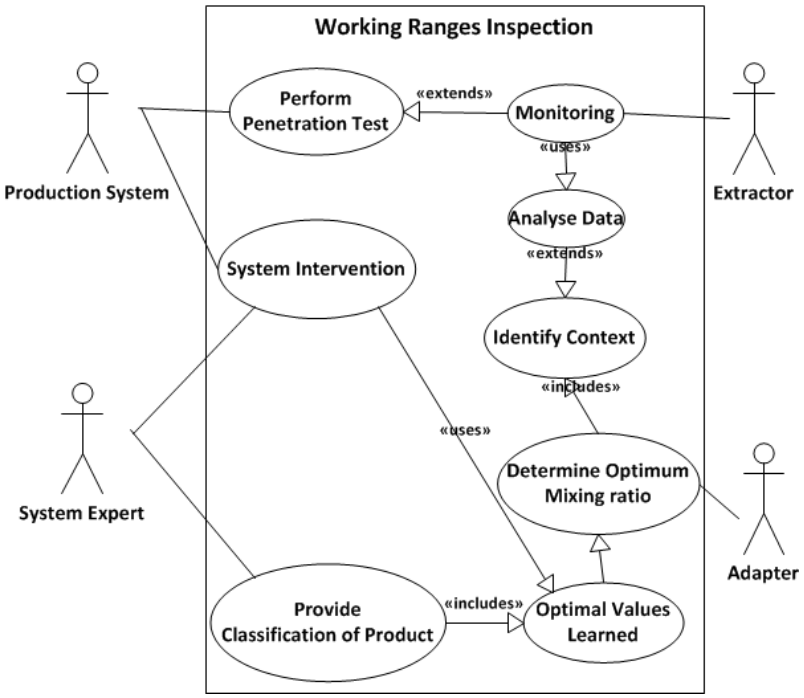
The figures 4.26(a) and 4.26(b) show an overview about the cup foam application scenario.

4.2.5.2.2 Application Scenario 2: Tank Refilling

After a tank filling, material may change the processing conditions on the machine (due to different temperatures of the material). As a matter of fact, a refilling with colder material reduces the temperature of the whole material in the tank, the viscosity increase. The higher viscosity causes a higher recirculation and injection pressure. If the injection pressure is too high and/or material temperature is to low, the injection stops and the article is scrap. Therefore an adaptation of the maximum and minimum control values to prevent the maximum injection pressure and minimum material temperature are required. Previously set limits for a certain production interval are not matching the current conditions; it could trigger errors during processing. Possible errors are for example the response of the injection pressure monitoring by a higher viscosity of a component, caused by a drop in temperature in the tank because the tank filling. Current situation is that the machine operator is adjusting manually machine parameters, when he is aware of changing conditions. DESMA intends to improve the tank refilling process by integrating SLPS solution into existent monitoring and control solution to be capable to react



(a)



(b)

Figure 4.26: Desma Cup Foam application scenario. (a) Penetration Depth adaptation. (b) Working Range Inspection.

on changing production/environment conditions. The automatic adjustment of machine parameters based on the changing (environmental) conditions, could minimize errors and keep the machine utilization high, as well as the overall product quality. Therefore the key issue in this application scenario is the intelligent monitoring of conditions and

automatic adjustment of machine parameters to guarantee a continuously high production quality.

The figure 4.27 shows an overview about the tank refilling application scenario.

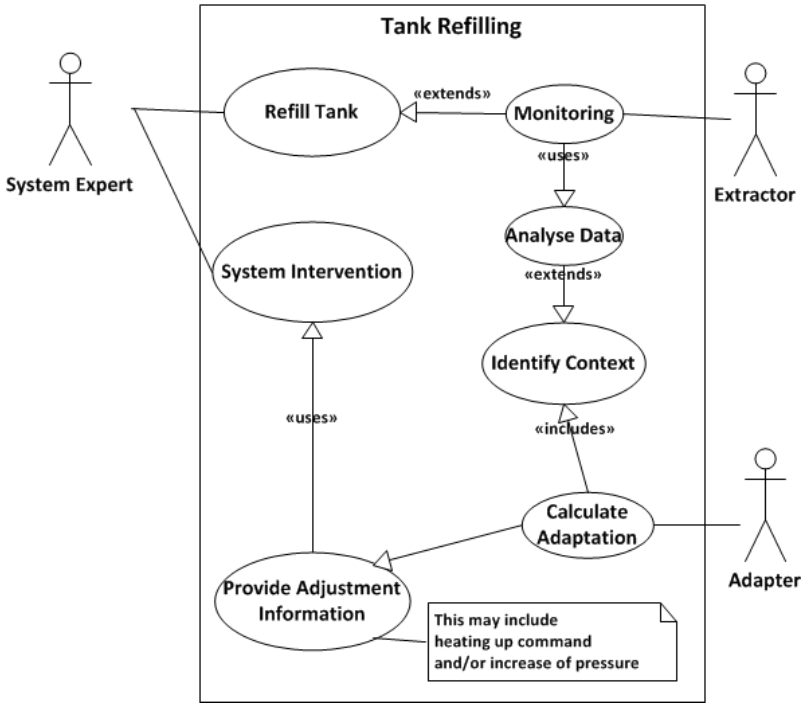


Figure 4.27: Desma Tank Refilling application scenario explanation

4.2.5.2.3 Application Scenario 3: Valve Synchronization

The synchronization of different valve circuits is a fundamental issue during the shoes production process, since several components have to be mixed without the premature advance of one of them to ensure a reproducible mixing quality. Today, the valve synchronization is performed by a patented mechanical synchronized control system that guarantees the perfect synchronicity between valves. However in some cases, the mechanical system cannot be installed has it is due to the lack of space in machines. As a result, differences in valve opening times can occur, caused by e.g. different force requirements, different air supply or valve abrasion, implying flaws in product quality. Today the synchronization is adjusted by a technician during downtime of the machines. DESMA intends to improve the valve synchronization by applying SLPS solution in order to implement an automatic adjustment of the valve switching to different conditions, by using intelligent monitoring. The monitoring should serve as a basis for identifying valve adjustment parameters. By having this automatic valve synchronization, a consistently high level of quality can be ensured. The second advantage of having an intelligent monitoring for the valve synchronization is that this can serve as a basis for preventive maintenance. Therefore the key issue in this application scenario is the intelligent monitoring of valve synchronization and automatic adjustment of the synchronization of

valves to assure product quality on the one hand and to achieve preventive maintenance of valves.

The figure 4.28 shows an overview about the tank refilling application scenario.

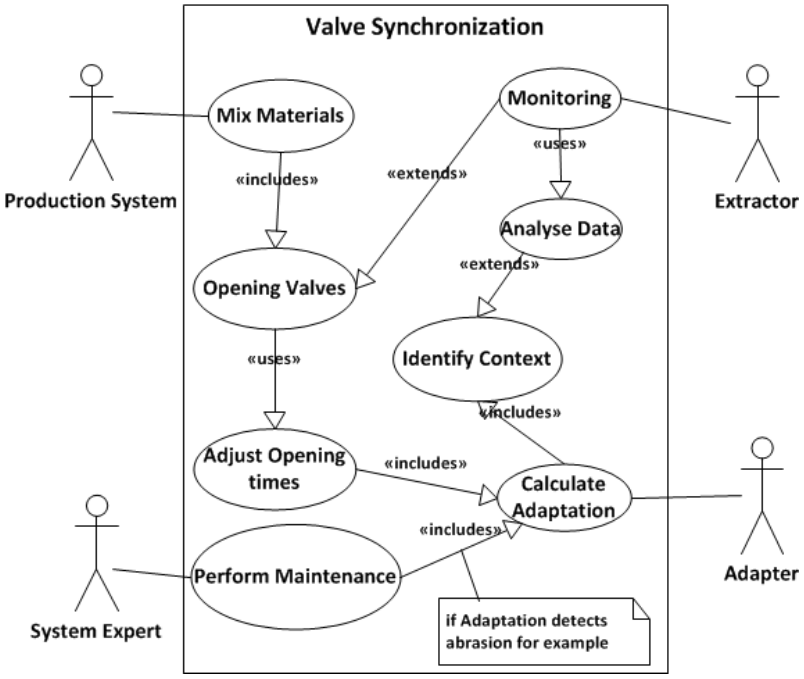


Figure 4.28: Desma Valve Synchronization application scenario explanation

4.2.5.2.4 Running the SLPS prototype

When starting the SLPS prototype the screen in figure 4.29 will be presented to the user:

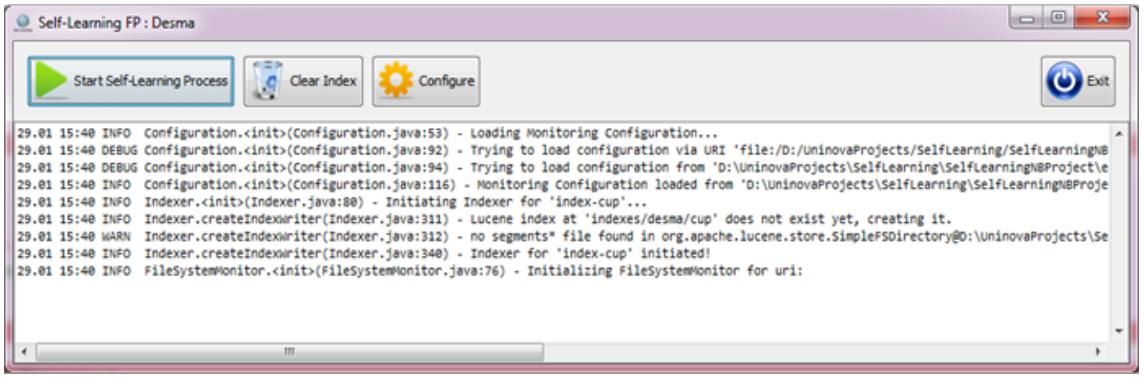


Figure 4.29: SLPS prototype Starter UI for Desma business case

After started the SLPS prototype using the *Start Self-Learning Process* button, the SLPS *Expert UI* appears and Self-Learning process is activated.

At this point it is important to state that all the UIs for the Desma application scenarios have been implemented by the ATB-Institute (Institute for Applied Systems

Technology Bremen GmbH), therefore only a brief description and some overview details, related essentially with the SLPS *Adapter*, will be presented in the following sections.

4.2.5.2.4.1 Cup Foam

The implemented SLPS prototype for the Desma Cup Foam application scenario focuses on calculating the optimum ratio between the the Polyurethen (A) and Polyisocyanat (B) components, and adjusting the process parameters in order to obtain the optimum mixing ratio to match the current context condition about "cup foam" process.

Whenever the Self-Learning process for the cup foam application scenario is activated, the following SLPS *Expert UI* (see figure 4.30) is presented to the system expert whenever the SLPS *Adapter* creates new suggestions for changing the mixing ratio.

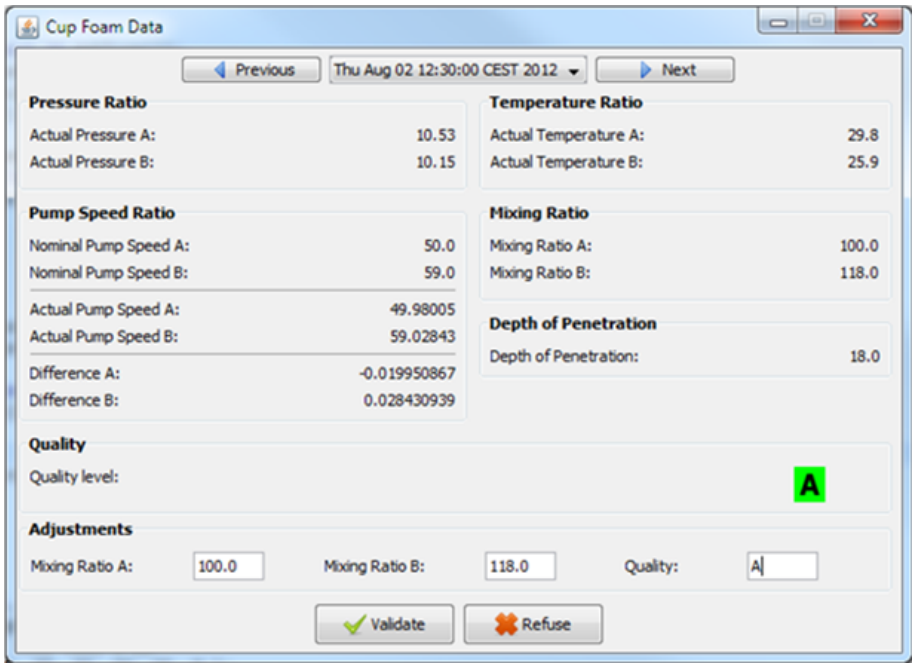


Figure 4.30: SLPS prototype Expert Collaboration UI for Desma Cup Foam application scenario

As mentioned before, usually two components are mixed together, therefore the SLPS *Expert UI* shows all the sensory information or in other words the values of the process parameters for component A and component B, that are used by the SLPS *Adapter* to determinate the optimum mixing ratio. Moreover, the SLPS *Adapter* is also capable to predict quality information about the produced shoe sole. The quality level information ranges from *A* (very good quality) to *C* (bad quality) and allows the classification of the data sets (the values of the monitored process parameters) in quality levels for describing the final product quality. Furthermore this information is also used to enable the system expert to analyze the working ranges of the considered process parameters and verify how they could influence the quality of the final product. For this reason the *Chart Foam*

Data UI (see figure 4.31) has been implemented by the ATB-Institute, showing all the information to the system expert to inspect the different working ranges of the monitored process parameters in order to identify the thresholds and, above all, the variable that causes the final product to be of poor quality.

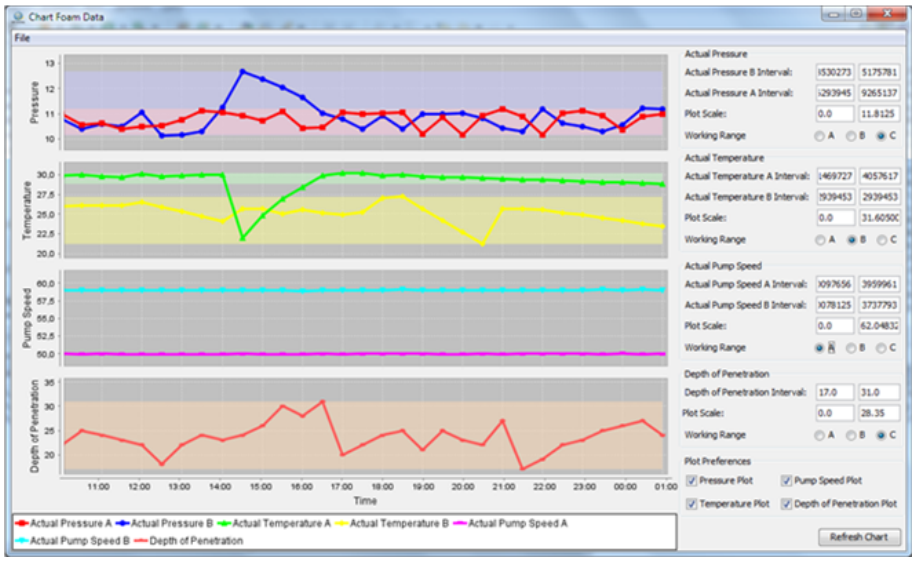


Figure 4.31: SLPS prototype Chart Foam Data UI for Desma Cup Foam application scenario

Finally, the operator can either validate (by clicking the *Validate* button) or refuse (by clicking the *Refuse* button) the adaptation displayed in the form below. Refusing the proposal will simply dismiss the window, whereas validation of the proposal will create a new adaptation file with the optimized mixing ratio values. This file, placed in the right directory, will be used by the Desma control system and update the mixing ratio for the next mixing process.

4.2.5.2.4.2 Tank Refilling

The implemented SLPS prototype for the Desma Tank Refilling application scenario focuses on adapting the production control system pressure and temperature threshold limits to face a punctual situation represented by the refilling of four tanks with new material components that are going to be used in a mixing process.

Whenever the Self-Learning process for the tank refilling application scenario is activated, the following SLPS *Expert UI* (see figure 4.32) is presented to the system expert whenever the SLPS *Adapter* creates new suggestions for adapting pressure and temperature limits.

The SLPS *Expert UI* gathers all the data from the context repository that has also been used by the SLPS *Adapter* during the adaptation process. The displayed data includes all the production parameters to frame the status of the tanks during the production activities and displays them to the system user expert in a graphical way. Furthermore, the SLPS *Adapter* suggestions about the new pressure and temperature limits related to each

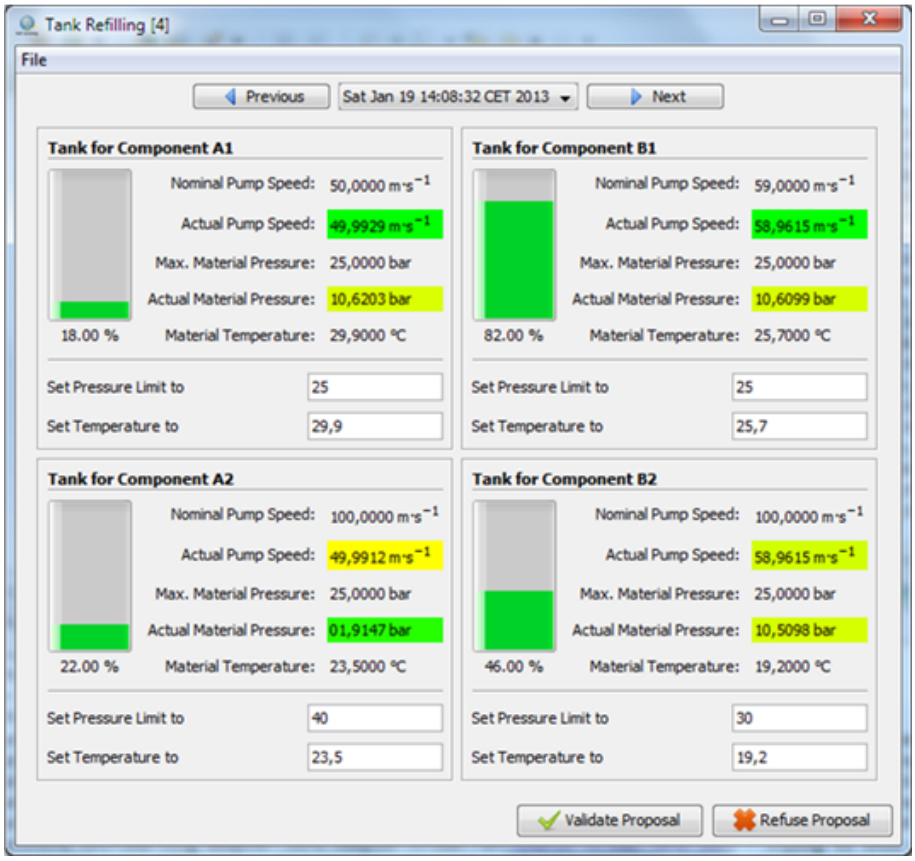


Figure 4.32: SLPS prototype Expert Collaboration UI for Desma Tank Refilling application scenario

displayed production context are also displayed, in such a way the system expert is capable to validate/refuse the adaptation proposals using the programmed button *Validate Proposal* and *Refuse Proposal*. If the values suggested by the SLPS *Adapter* are validated, the generated adaptation will be parsed from the Desma control system in order to apply the adjustments. Furthermore, the system expert can override the suggested pressure and temperature values and finally validate the proposal, than the learning model of the process (stored into the *data-to-learn* text file) will be updated with the new available knowledge from the system expert. On the contrary, if the system expert decides to refuse the proposed values completely the adaptation will not be sent to the Desma system and the learning models of the process will remain untouched.

Finally, in the context of Desma Tank Refilling application scenario the *Model Data Viewer* functionality (already seen in section 4.2.5.1.2) is available but not used.

4.2.5.2.4.3 Valve Synchronization

The implemented SLPS prototype for the Desma Valve Synchronization application scenario focuses on adjusting the opening times of several valves attached to a so called *mixing head* based on the identified context in order to ensure a reproducible product quality.

Whenever the Self-Learning process for the valve synchronization application scenario is activated, the following SLPS *Expert UI* (see figure 4.33) is presented to the system expert whenever the SLPS *Adapter* creates new suggestions for adjusting the opening times for the valves.

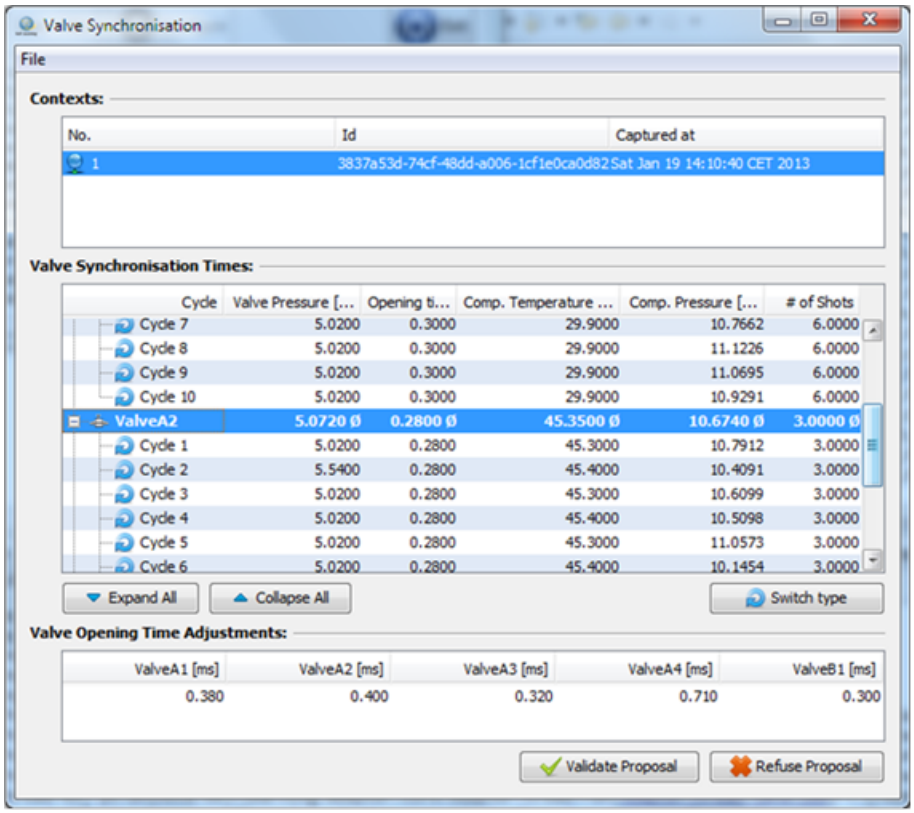


Figure 4.33: SLPS prototype Expert Collaboration UI for Desma Valve Synchronization application scenario

The SLPS *Expert UI* gathers all the data from the context repository that has also been used by the SLPS *Adapter* during the adaptation process. The displayed data includes all the production parameters that can directly and/or indirectly affect the opening times of the valves and consequently their synchronicity and displays them to the system user expert. The SLPS *Expert UI* is organized in three parts: *Contexts*, *Valve Synchronisation Times* and *Valve Opening Time Adjustments*. The *Contexts* part shows the identifier (*id*) and the time stamp (*Captured at*) of the detected contexts. The *Valve Synchronisation Times* part details all the information about the opening times of the valves as well as sensory information about pressure and temperature gathered from the Desma production system. The *Valve Opening Time Adjustments* shows the SLPS *Adapter* predictions about opening times of the valve based on the current context and the process knowledge gathered during the system lifecycle, the predictions will be used by the system expert to adjust the opening times and ensuring the valves synchronicity. If the values suggested by the SLPS *Adapter* are validated, the generated adaptation will be parsed from the Desma control system in order to apply the adjustments. Furthermore, the system expert can override

the suggested opening times values for each valve and finally validate the proposal, than the learning model of the process (stored into the *data-to-learn* text file) will be updated with the new available knowledge from the system expert. On the contrary, if the system expert decides to refuse the proposed values completely the adaptation will not be sent to the Desma system and the learning models of the process will remain untouched.

Finally, in the context of Desma Valve Synchronization application scenario the *Model Data Viewer* functionality (already seen in section 4.2.5.1.2) is available but not used.

4.2.5.2.5 Experimental Results

To validate current SLPS platform fitness and performance, the proposed SLPS solution has been integrated into real industrial equipment and used to identify production process operative context and react to changing situations associated with variations in different parameter sets in order to improve error-prone processes (caused by humans) and reduce maintenance problems. To do this, the SLPS solution has been fed with a set of optimum manufacturing process parameters gathered by observing the production process. These set of parameters are then used by the SLPS *Adapter* to build a representative model of process relying on empirical data. The parameters considered to build the model are the pressure and the temperature, speed frequencies of drives and pumps, proper material mix ratio and filling of materials into shoe forms.

First application of the presented approach in real world scenarios is pointing at promising results. The application in control systems/machines and automation systems for shoe industry documented that the objective to enhance machines with self-learning functionalities to keep the process parameters always inside the optimum working range were fulfilled. Therefore, implementation of the proposed SLPS solution for the automatic adjustment of machine parameters based on changing context, for example changing ambient conditions, leads to minimization of errors and keeps the machine utilization high, as well as the overall product quality. Moreover, the SLPS solution experiments have proved to fulfill reliability, feasibility and robustness requirements in a complex industrial setting enabling its integration into a real industrial environment. At this stage, Desma confirmed the potentials of the self-learning approach and methodology and is intended to carry out further research in order to integrate the SLPS solution into their products.

4.2.5.3 Business Case 3 at Fastems: Self-Learning dynamic scheduling and dispatching in Flexible Manufacturing Systems (FMS) for automotive industry

4.2.5.3.1 Application Scenario: Dynamic Scheduling and Dispatching for Flexible Manufacturing Systems (FMS)

The optimum management of FMS resources is fundamental in the field of industrial production for decreasing production costs. In general, each customer has his own idea of optimal operations when using FMS, and is strictly related with the production objectives

or profile. According to the production objectives one or a combination of the following optimization criteria can be selected:

- Maximum utilization rate of the production machines
- Minimize lead time of production orders
- Keeping the due delivery dates of production orders
- Minimize the tool flow in production machines

These optimization criteria are normally not static since they are changing depending on both the production profile and on the process state. For instance, when most of the production load is addressed for direct customer orders it is necessary to keep the due delivery dates of the production orders as optimization criteria. When most of the production is for *Kanban* manufacturing (stock batches) then it is natural to try to maximize the machine utilization rate. In this context the scheduling and dispatching of the resources of a FMS assumes a key importance to improve the performance of the FMS. As exposed in chapter 3, the performance of a FMS not supported by an efficient scheduling and dispatching of the resources drastically reduce the advantages derived from its flexibility. The Fastems application scenario grounds on the deep use of SLPS solution to improve the reactive/dynamic scheduling model for enhancing general performance of the FMS by

- taking into account the operator supervision concerning the optimization criteria;
- dynamically changing its internal job priority rule depending on the process context in which the FMS is operating and operator supervision and learning from them;
- and introducing resource planning features.

Although Fastems uses a reactive scheduling approach the problem is that there are no single optimal priority rule since customers have different conceptions of what is optimal and one rule does not provide optimal result in all process situations. Therefore, Fastems intends to apply SLPS solution together with the existing monitoring and control platform for gathering all the necessary information about the manufacturing production process as well as input from human expert experience concerning the optimization criteria. The SLPS solution will process all the amount of gathered data to dynamically adapt the scheduling plans for avoiding loading stations starvation while improving maximum machines utilization. The general flow of operation between Fastems shop floor and SLPS solutions is shown in figure 4.34.

Fastems has developed SOA based control system architecture (*Manta architecture*) that is based on Microsoft .NET platform. *Manta* based control system comprises a set of typed services associated to the manufacturing resources through an identifier (for example each machine tool has its own service). These services will be used by the SLPS solution to interact with the Fastems shop floor.

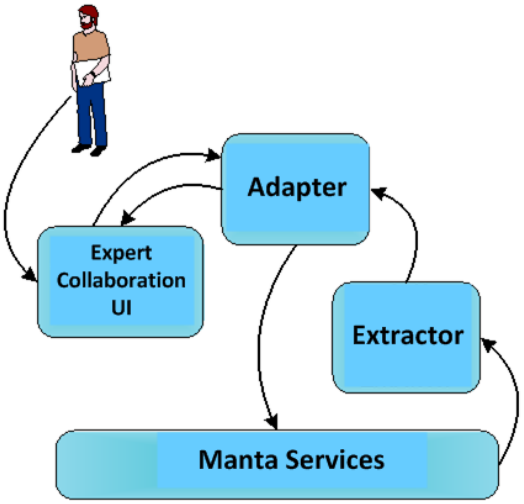


Figure 4.34: General flow of operation between SLPS solution and Fastems shop floor [Self-Learning, 2010c]

4.2.5.3.2 Running the SLPS prototype

The implemented SLPS prototype for the Fastems application scenario focuses on two main tasks, namely extraction of necessary contextual information about the production status of the FMS resources from Fastems shop floor and the selection of the best rule to apply according to the contextual information and the entire system lifecycle. The extraction of the contextual information as well as the communication of the best rule to apply is performed using the provided *Manta services*.

When starting the SLPS solution the screen in figure 4.35 will be presented to the user:

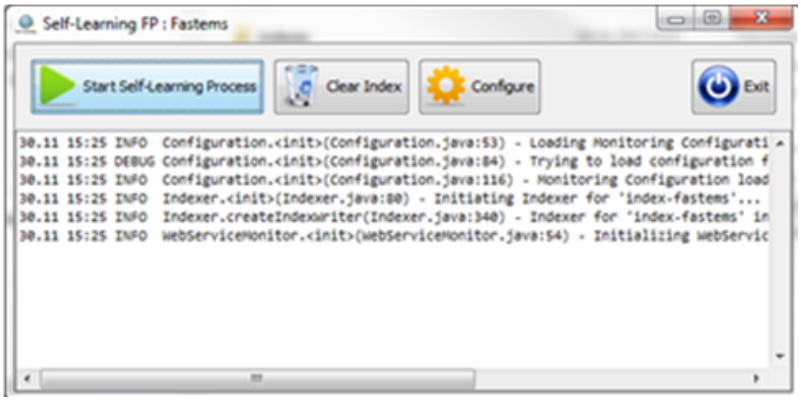


Figure 4.35: SLPS prototype Starter UI for Fastems business case

After started the SLPS prototype using the *Start Self-Learning Process* button, the SLPS *Expert UI* (see figure 4.36) appears and Self-Learning process is activated.

Since the SLPS *Expert UI* for the Fastems application scenario has been in part implemented by the TUT (Tampere University of Technology) then only the main functionalities related essentially with the SLPS *Adapter* will be presented.

The SLPS *Expert UI* for the Fastems application scenario, is composed by two tables:

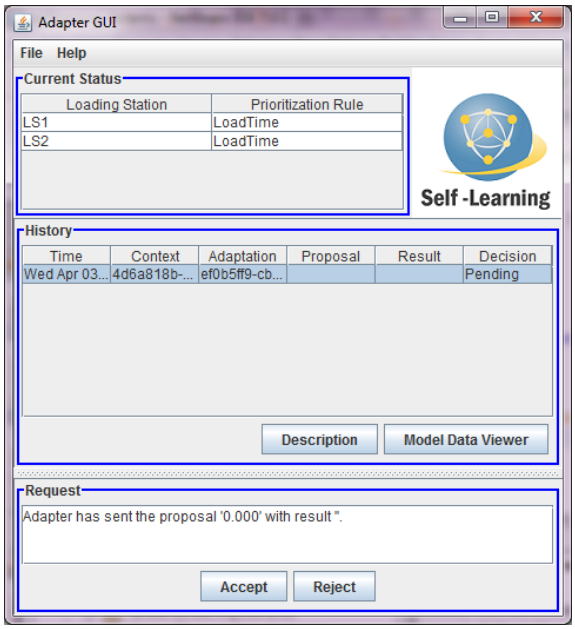


Figure 4.36: SLPS prototype Expert Collaboration UI for Fastems business case

Current Status and *History* tables. The *Current Status* table is used to show the current status of the Fastems FMS, i.e. the loading stations available and the prioritization rule selected. The *History* table is used to show the adaptation proposal calculated by the SLPS *Adapter*. Therefore, whenever a new *Adaptation* is sent by the SLPS *Adapter* to the SLPS *Expert UI* it will be shown in the *History* table and the following details will be presented, namely the *Time* field that shows the time stamp, the *Context* field that contains the identifier of the context that triggered the adaptation proposal, *Adaptation* field that contains the the identifier of the *Adaptation* itself, the *Proposal* field that contains the rule that best fits the current context, the *Result* field that contains the final user expert decision and the *Decision* that shows the status of the current *Adaptation* (Accepted, Refused or Pending). Moreover, to help the user expert during the validation task the two buttons *Description* and *Model Data Viewer* have been implemented. The former can be used to contextualize the current adaptation proposal showing all the information about the context that has triggered the *Adaptation* (see figure 4.37). The latter is used to retrieve the result of the *X-Cross* validation performed automatically by the SLPS *Adapter* during its lifecycle in order to have an idea on the accuracy of the SLPS *Adapter* proposals when tested on new unseen situations. The *Model Data Viewer* button has the same functionalities and features of the *Model Data Viewer* button that has been presented in the section 4.2.5.1.2 and thus used to show the *Model Viewer* and the *Model Evolution Viewer* forms.

Finally, the *Accept* button is used to both validate the adaptation proposal as it is or eventually modified by the user expert of the system. Whenever it is pressed the selected prioritization rule is sent to the Fastems platform through the *Manta services* and then to the SLPS platform for updating the related *data-to-learn* text file. On the contrary, the *Reject* button is used to refuse the adaptation proposal implying that the SLPS *Adapter*

Figure 4.37: SLPS prototype Proposal Description

suggestion about the prioritization rule will not be sent to the Fastems platform and then not be be used by the SLPS platform for updating the learning model of the process (*data-to-learn* text file) and, thus, not considered for future adaptation proposals.

4.2.5.3.3 Learning Algorithm

In this scenario, the main objectives are maximizing the machine utilization rate and avoiding loading stations starvation. Based on the information about the state of the shop floor environment and on the operator's optimization criteria, the SLPS system has to be able to dynamically choose a rule, inside a set of pre-programmed rules, that simultaneously satisfies the two main objectives. The problem can be simply modelled as a set of rules that suggests the utilization of the *Rule Induction* algorithm. Others algorithms can be successfully applied to this scenario such as *Naïve-Bayes* and *ID3* learners.

4.2.5.3.4 Experimental Results

To validate current platform fitness and performance, Fastems *FPM DemoLite* simulator was used. This tool is a table top application containing full-scale FMS control system with emulated process devices (RGV, loading stations, machine tools) and simulates the complete behavior of a FMS cell while offering an essential mean to test different production contexts. To test the Fastems application scenario the configuration presented in figure 4.38 has been implemented.

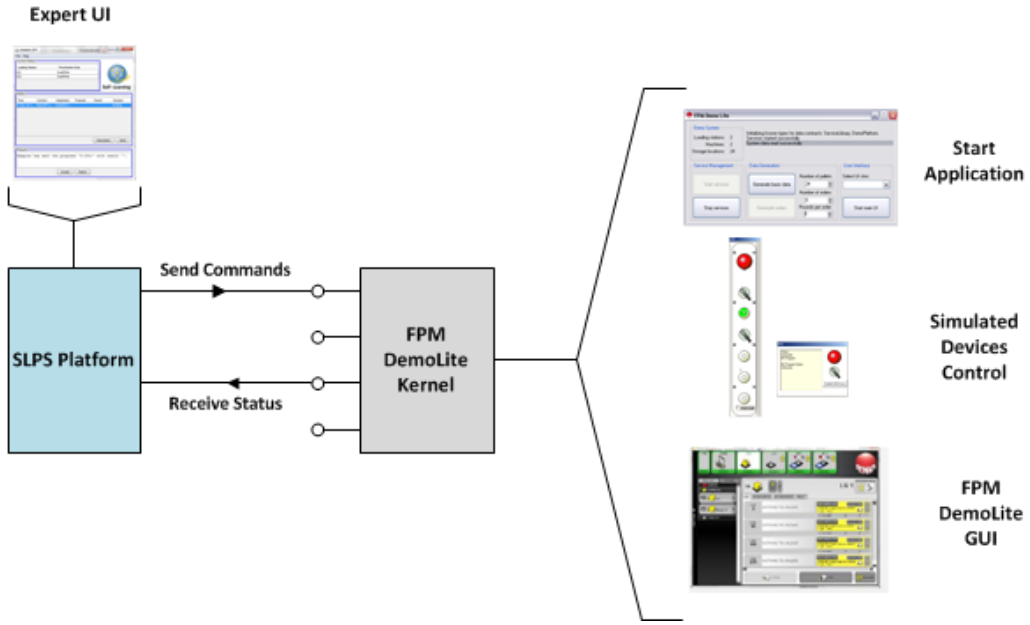


Figure 4.38: Implementation platform for Fastems application scenario [Self-Learning, 2010c]

The SLPS platform has been connected to the *FPMDemoLite* using special web-services provided by the *FPMDemoLite* kernel. Furthermore, the *FPMDemoLite* will provide a set of UIs that can be used to configure, start and run the FMS environment. In this scenario,

common Fastems experts heuristics were followed to determine the appropriate behavior of the system for each context. The objective is to ensure that adaptation proposals converge to a level of accuracy that after some iterations will allow the system to run itself without any system expert intervention, learning from previous contextual decisions. The explicit process model has been dynamically constructed considering the following process parameters provided by Fastems system expert: total loading and machining times, current priority mode, number of pallets, highest, lowest and average priorities.

The following steps are necessary to prepare the testbed:

- Start the *FPMDemoLite* application.
- Start the SLPS application and the SLPS Expert UI.
- Activate the manufacturing using the *FPMDemoLite* GUI.

At the beginning of the production process the machines are empty but there are loading queue (see figure 4.39) implying that, to get the orders/pallets running, the best rule to choose is *LoadTime* priority.

Figure 4.39: Manufacturing process status at the beginning of the production

The SLPS solution in this phase is not able to suggest any rule since its internal learning model of the process is empty, thus no proposal are shown at all. However, after a transient phase where the system expert selects the correct rule (*LoadTime*) and confirms gradually the SLPS *Adapter* learns with system expert decisions, or in other words improve its knowledge about the manufacturing process and starts to suggest the correct rule to apply for the current context (see figure 4.40).

Moreover, while advancing the production process the orders/pallets have been loaded into the related stations and machining is in progress (see figure 4.41). In this context the best rule to chose is the *OrderPriority* in order to feed the machining stations with the orders/pallets with higher priority.

History

Time	Context	Adaptation	Proposal	Result	Decision
Tue Feb 05 ...	ce32c4a1-6...	474dfabc-a...		LoadTime	Accepted
Tue Feb 05 ...	421b6d51-0...	5c97ab3f-7...		LoadTime	Accepted
Tue Feb 05 ...	1472124b-c...	e00dd959-5...		LoadTime	Accepted
Tue Feb 05 ...	fa23a22c-b...	5c2e4cd8-5...		LoadTime	Accepted
Tue Feb 05 ...	d46eff7d-7f...	5f640548-3...	LoadTime	LoadTime	Accepted
Tue Feb 05 ...	a8ff3fa2-fd...	d6a09afe-b...	LoadTime	LoadTime	Accepted
Tue Feb 05 ...	1d9999cc-af...	76bac5e1-b...	LoadTime	LoadTime	Accepted
Tue Feb 05 ...	b151c123-f...	1f2547ff-58...	LoadTime	OrderPriority	Accepted
Tue Feb 05 ...	0358b6c3-7...	5a6b3eba-a...		OrderPriority	Accepted

Figure 4.40: SLPS *Adapter* proposals at the beginning of the production

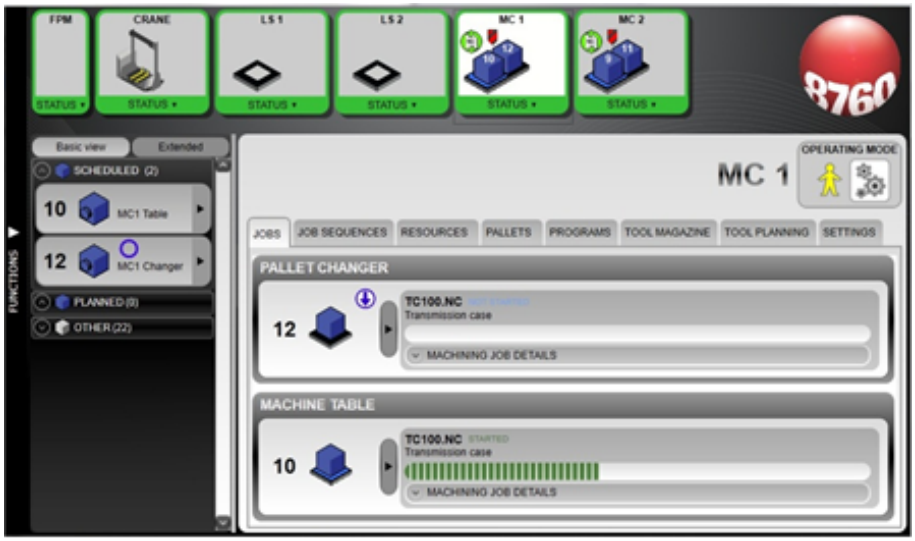


Figure 4.41: Manufacturing process status while advancing the production

The SLPS solution in this new operative context continues to suggest the *LoadTime* priority rule but the system expert corrects its behaviour changing the rule to the *OrderPriority*. The SLPS *Adapter* learns with system expert decisions and after very few interactions it starts to suggest the correct rule to apply to this new operative context (see figure 4.42).

Tue Feb 05 ...	a8ff3fa2-fd...	d6a09afe-b...	LoadTime	LoadTime	Accepted
Tue Feb 05 ...	1d9999cc-af...	76bac5e1-b...	LoadTime	LoadTime	Accepted
Tue Feb 05 ...	b151c123-f...	1f2547ff-58...	LoadTime	OrderPriority	Accepted
Tue Feb 05 ...	0358b6c3-7...	5a6b3eba-a...		OrderPriority	Accepted
Tue Feb 05 ...	947d98f6-1...	fdc341b7-7...	OrderPriority	OrderPriority	Accepted
Tue Feb 05 ...	f4d9f15c-46...	b931660b-3...	OrderPriority	OrderPriority	Accepted
Tue Feb 05 ...	986844f6-9...	09207980-4...	OrderPriority	OrderPriority	Accepted
Tue Feb 05 ...	111e4f5e-5...	462bb099-1...	LoadTime	LoadTime	Accepted
Tue Feb 05 ...	3399d814-a...	da05fc06-6c...	LoadTime	LoadTime	Accepted
Tue Feb 05 ...	635866b3-b...	89cf5687-d...	LoadTime	LoadTime	Accepted
Tue Feb 05 ...	3de566b5-5...	e1c3f90e-a...	LoadTime	LoadTime	Accepted
Tue Feb 05 ...	8a28a494-5...	ece4daa0-2...	LoadTime	LoadTime	Accepted

Figure 4.42: SLPS *Adapter* proposals while advancing the production

Continuing the production process, the SLPS *Adapter* starts to make reliable proposals

while learning to detect when it is time to change the priority rule. In this case, the system expert not need to change any adaptation proposal, and the SLPS *Adapter* automatically detects when rule need to be changed (see figure 4.43).

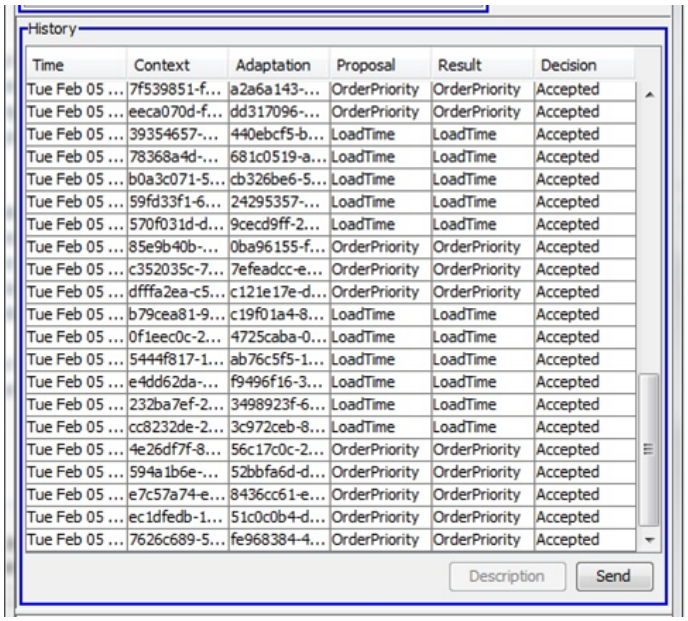


Figure 4.43: SLPS *Adapter* proposals after the transient phase

Finally, the figure 4.44 presents the early accuracy results obtained with 83 cross validation tasks performed during system operation. It is visible in the graph that the accuracy improves over time (in proportion to the information added into the learning model coming from adaptations performed to the system) and tends to stabilize around an accuracy of 80%. Further tests will be performed on real equipment to confirm these values and, if necessary, correct and optimize some of the learning parameters.

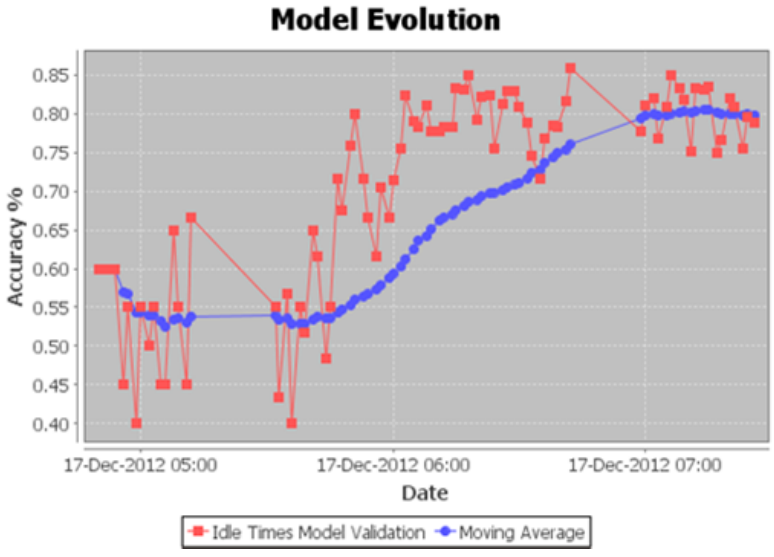


Figure 4.44: Early experimental cross validation results

5

Conclusions and Future Work

5.1 Conclusions

Manufacturing companies are today under daily pressure to sustain positive growth, rapidly introduce new and customized products, earn competitive advantage in market sharing, and satisfy the new environmental regulations requirements while reducing the production costs without affecting the final product quality. To accomplish this, manufacturing companies are engaged in an innovation race to implement more and more exclusive and efficient production systems. As a result, several manufacturing processes, based on the most diverse technologies, architectures, approaches and methodologies, have been designed and implemented through the years by researches and practitioners to satisfy mass customization requirements. Some manufacturing processes are focused on the improving of responsiveness, reconfigurability and lead time; while others are focused on the improving of the final product quality, optimization of the production activities, waste elimination, integration of secondary processes in the main control, improving the visibility inside the manufacturing companies facilitating the information flow between all the layers of a manufacturing company. The Self-Learning Production System paradigm falls in this second category intending to deliver additional productivity gains to a manufacturing process extending the reach of the automation system beyond the world of process control considering maintenance and energy efficiency without affecting traditional control approaches. Within the framework of manufacturing production systems, Self-Learning concept proposes the application of machine learning techniques to allow computer based control systems to change their own behavior based on the lifecycle information or, more broadly, on the knowledge and patterns extracted from all the available data. In this context, the presented dissertation focus on a new approach that exploits fundamental data generated during the production process in order to learn and

predict the behaviour of the manufacturing process. The predictions are then used to support the manufacturing process sustainability in terms of energy saving, easy adaptation of the manufacturing process to the companies specific optimization objectives and final product quality by computing the better set of manufacturing process parameters. This data exploitation provides a step forward to answer the elemental question of how to use the huge amount of manufacturing data to know more about the process.

The idea of applying data mining techniques to the manufacturing context is not recent as exposed in the chapter 2, however the implemented solutions have not been wide applied into real industrial environment. Nowadays, the context evolved into inexpensive hardware solutions that together with the numerous free software tools available on the Internet open the doors to the propagation of this techniques to the manufacturing context. Moreover, the wide dissemination of industrial standards for monitoring and control manufacturing processes if from one side allows to cope with the high complexity of actual manufacturing production lines, providing a set of well-defined rules to design and implement control and monitoring solutions to guarantee the correct execution of all the steps of production cycle bringing to the creation of goods, on the other side limit the dissemination of new and powerful solutions and approaches for design and develop monitoring and control solutions to cope with the current challenges. In this scenario, the Self-Learning approach intends to provide the capability to improve and enhancing the current monitoring and control solutions without affecting the way the systems are designed and developed. The usage of the SOA paradigm together with the Web-Services technology provides a completely new way of building applications within a more powerful, flexible and easy-to-integrate programming model. All these aspects improve the trust and the acceptance of the general SLPS solution by industrial world.

Current work presents an important contribution to the SLPS research domain by providing a fully functioning solution able to improve the overall system performance and applicable to different manufacturing systems and processes. The improvements of manufacturing system performance are obtained by a more intelligent use of all the manufacturing data produced during the production process, while the applicability to different production contexts and/or different layer inside the manufacturing company is guaranteed by the modular and abstract approach used for designing and implementing the SLPS architecture.

The business cases and their related application scenarios presented in chapter 4 proved the genericity of the SLPS solutions since completely distinct manufacturing processes as well as process specific goals and objectives are considered. The developed SLPS solution has been integrated within these application scenarios with little programming effort. Furthermore during the experimental stage, the SLPS solution was capable to process the manufacturing execution data, compute and infer useful conclusions regarding the process, present the results to the system expert for validation, and most important learn with system expert decisions as a foundation for consistent evolvable behaviour along time. Therefore, even the system has proved to work autonomously, the

system expert continues to play a fundamental role inside the SLPS lifecycle for actively cooperating with it during the validation phase. Profiting from these characteristics, the application of the SLPS solution and in particular the integration into the manufacturing context of the SLPS *Adapter*, i.e. of a generic component capable to analyze the huge amount of data produced during the execution of the production process, can really enhance traditional monitoring and control solutions. As a matter of fact, the presented application scenarios confirm an increasing of the energy efficiency when working with machines tools, an improvement of the final product quality as well as a better reactiveness of the dynamic scheduling in the context of FMS. Furthermore, other facets of a production process can be analyzed simply changing the set of manufacturing process parameters. Moreover, interactions with system expert, fundamental in the initial stage to define the SLPS *Adapter* behaviour, are minimized along time showing the capability of the SLPS *Adapter* to emulate the human behaviour. However, the other side of the coin is that a wrong choice of the manufacturing process parameters can led to unuseful and incorrect suggestions due to the inability to discriminate different contexts. Furthermore, wrong system expert decisions during SLPS lifecycle about can also led to incorrect suggestions. Therefore, the SLPS *Adapter* behaviour strictly depends on the quality of the generated learning model that in turn depends on both the quality of the manufacturing production process parameters and the quality of system expert suggestions implying that if they are contaminated the entire SLPS *Adapter* behaviour is undermined.

These consideration are the foundation for the following section 5.2.

5.2 Future Work

The work described in this dissertation provides a feasible SLPS solution, all the results shown are easily and completely reproducible, however several future developments can be also considered and added to the current prototype in order to further improve its robustness and quality of the adaptation proposals. In this direction, configurations tools can be implemented and added to the current prototype in order to allow the system expert to configure/clean the learning models of the process, to select which learning model to use, to allow dynamic configuration of the *Adapter Proactive Behaviour* parameters. An optimization for product-level deployments is needed in the SLPS *Adapter* Java code. Finally, next to this auxiliary tools, several improvements can be implemented for the SLPS *Learning Module*, *in primis* the migration from the RapidMiner 4.6 API to the new and completely different RapidMiner 5.0 API is peremptory but not trivial since the entire RapidMiner architecture has been redesigned. Furthermore, until now only supervised machine learning techniques are explored implying that other learning techniques could be considered such as supervised learning opening the doors to new potential application scenarios. Considering all these entries, the following questions are worth exploring:

- How to implement/integrate a process configuration tools for allowing the system expert to optimize the SLPS prototype to the particular application scenario?

- How to introduce other learning techniques such as unsupervised learning techniques in the solutions?
- is it possible to use different kind of learning algorithm for the same problem?
- How to implement an online validation mechanism for the learning models?
- How to implement a supporting tool for helping during the selection of the manufacturing process parameters?
- How to improve the Adapter Proactive Behaviour considering more proactivity dimensions and not only the counter level and the time-out level?

5.3 Scientific Contributions

The work done under the scope of the Self-Learning project by the author for designing and implementing the SLPS Adapter component, and widely described into this document, resulted in a set of scientific contributions that have been published in several conferences, namely:

- Cândido, G., Di Orio, G., Barata, J., & Scholze, S. (2012). Adapter for self-learning production systems. Technological innovation for value creation, 171-178.
- Cândido, G., Di Orio, G., Barata, J., & Bittencourt, J., Bonefeld, R. (2013). Self-Learning Production Systems (SLPS) - Energy Management Application for Machine Tools. Proceedings of the 22nd IEEE International Symposium on Industrial Electronics, ISIE 2013 (to appear).
- Cândido, G., Di Orio, G., Barata, J. (2013). Self-Learning Production Systems (SLPS) - Adapter Reference Architecture. Proceedings of the 23rd International Conference on Flexible Automation and Intelligent Manufacturing, FAIM 2013 (to appear).
- Di Orio, G., Cândido, G., Barata, J., & Scholze, S., Kotte, O., Stokic, D. (2013). Self-Learning Production Systems (SLPS) - Optimization of Manufacturing process parameters for the Shoe Industry. Proceedings of the 11th IEEE International Conference on Industrial Informatics, INDIN 2013 (to appear).
- Di Orio, G., Cândido, G., Barata, J., & Scholze, S., Kotte, O., Stokic, D. (2013). Self-Learning approach to support lifecycle optimization of Manufacturing processes. Proceedings of the 39th Annual Conference of the IEEE Industrial Electronics Society, IECON 2013 (to appear).
- Di Orio, G., Cândido, G., Barata, J., & Bittencourt, J., Bonefeld, R. (2013). Energy Efficiency in Machine Tools - A Self-Learning Approach. Proceedings of the 2013 IEEE Iternational Cconference on Systems, Man, and Cybernetics, SMC 2013 (submitted).

Bibliography

[iso, 2001] (2001). ISO/IEC 9126-1 - software engineering — product quality.

[Ahmed et al., 2010] Ahmed, N., Atiya, A., El Gayar, N., and El-Shishiny, H. (2010). An empirical comparison of machine learning models for time series forecasting. *Econometric Reviews*, 29(5-6):594–621.

[Alexopoulou et al., 2009] Alexopoulou, N., Kanellis, P., Nikolaidou, M., and Martakos, D. (2009). A holistic approach for enterprise agility. *Handbook of Research on Enterprise Systems*.

[Alpaydin, 2004] Alpaydin, E. (2004). *Introduction to machine learning*. MIT press.

[Ansoff, 2006] Ansoff, H. (2006). Strategic issue management. *Strategic Management Journal*, 1(2):131–148.

[Babiceanu and Chen, 2006] Babiceanu, R. and Chen, F. (2006). Development and applications of holonic manufacturing systems: a survey. *Journal of Intelligent Manufacturing*, 17(1):111–131.

[Barata, 2005] Barata, J. (2005). *Coalition Based Approach For ShopFloor Agility*. Orion, Amadora - Lisboa.

[Barata et al., 2006] Barata, J., Santana, P., and Onori, M. (2006). Evolvable assembly systems: a development roadmap.

[Barry, 2003] Barry, D. (2003). *Web services and service-oriented architecture: the savvy manager's guide*. Morgan Kaufmann Pub.

[Batanov et al., 1993] Batanov, D., Nagarur, N., and Nitikhunkasem, P. (1993). Expertmm: A knowledge-based system for maintenance management. *Artificial intelligence in engineering*, 8(4):283–291.

[Bell, 1962] Bell, D. (1962). *The end of ideology: on the exhaustion of political ideas in the fifties: with" The resumption of history in the new century"*. Harvard University Press.

[Bellwood et al., 2002] Bellwood, T., Clément, L., Ehnebuske, D., Hately, A., Hondo, M., Husband, Y., Januszewski, K., Lee, S., McKee, B., Munter, J., et al. (2002). The universal description, discovery and integration (uddi) specification.

[Bengtsson and Dabhilkar, 2009] Bengtsson, L. and Dabhilkar, M. (2009). Manufacturing outsourcing and its effect on plant performance—lessons for kibs outsourcing. *Journal of evolutionary economics*, 19(2):231–257.

[Bishop, 2006] Bishop, C. (2006). Pattern recognition and machine learning (information science and statistics).

[Bittencourt et al., 2011] Bittencourt, J., Bonefeld, R., Scholze, S., Stokic, D., Uddin, M., and Lastra, J. (2011). Energy efficiency improvement through context sensitive self-learning of machine availability. In *Industrial Informatics (INDIN), 2011 9th IEEE International Conference on*, pages 93–98. IEEE.

[Blum, 2007] Blum, A. (2007). Machine learning theory.

[Box et al., 1999] Box, D., Ehnebuske, D., Kakivaya, G., Layman, A., Mendelsohn, N., Nielsen, H. F., Thatte, S., and Winer, D. (1999). Soap: Simple object access protocol. *HTTP Working Group Internet Draft*.

[Bray et al., 1997] Bray, T., Paoli, J., Sperberg-McQueen, C. M., Maler, E., and Yergeau, F. (1997). Extensible markup language (xml). *World Wide Web Journal*, 2(4):27–66.

[Browne et al., 1988] Browne, J., Harhen, J., and Shivnan, J. (1988). *Production management systems: a CIM perspective*. Addison-Wesley England.

[Cachapa et al., 2007] Cachapa, D., Colombo, A., Feike, M., and Bepperling, A. (2007). An approach for integrating real and virtual production automation devices applying the service-oriented architecture paradigm. In *Emerging Technologies and Factory Automation, 2007. ETFA. IEEE Conference on*, pages 309–314. IEEE.

[Camarinha Matos et al., 1995] Camarinha Matos, L., Pinheiro-Pita, H., Rabelo, R., and Barata, J. (1995). Towards a taxonomy of CIM activities. *International Journal of Computer Integrated Manufacturing*, 8:160 — 176.

[Cândido et al., 2009] Cândido, G., Barata, J., Colombo, A., and Jammes, F. (2009). SOA in reconfigurable supply chains: A research roadmap. *Engineering applications of artificial intelligence*, 22(6):939–949.

[Cândido et al., 2011] Cândido, G., Colombo, A. W., Barata, J., and Jammes, F. (2011). Service-oriented infrastructure to support the deployment of evolvable production systems. *Industrial Informatics, IEEE Transactions on*, 7(4):759–767.

[Cannata et al., 2008] Cannata, A., Gerosa, M., and Taisch, M. (2008). A technology roadmap on soa for smart embedded devices: Towards intelligent systems in manufacturing. In *Industrial Engineering and Engineering Management, 2008. IEEM 2008. IEEE International Conference on*, pages 762–767. IEEE.

[Carbonneau et al., 2008] Carbonneau, R., Laframboise, K., and Vahidov, R. (2008). Application of machine learning techniques for supply chain demand forecasting. *European Journal of Operational Research*, 184(3):1140–1154.

[Channabasavaiah et al., 2003] Channabasavaiah, K., Holley, K., and Tuggle, E. (2003). Migrating to a service-oriented architecture. *IBM DeveloperWorks*, 16.

[Charaniya et al., 2010] Charaniya, S., Le, H., Rangwala, H., Mills, K., Johnson, K., Karypis, G., and Hu, W. (2010). Mining manufacturing data for discovery of high productivity process characteristics. *Journal of biotechnology*, 147(3):186–197.

[Chen, 2003] Chen, M. (2003). Configuration of cellular manufacturing systems using association rule induction. *International Journal of Production Research*, 41(2):381–395.

[Cherkassky and Mulier, 2007] Cherkassky, V. and Mulier, F. (2007). *Learning from data: concepts, theory, and methods*. Wiley-IEEE Press.

[Chien et al., 2007] Chien, C., Wang, W., and Cheng, J. (2007). Data mining for yield enhancement in semiconductor manufacturing and an empirical study. *Expert Systems with Applications*, 33(1):192–198.

[Christensen et al., 2001] Christensen, E., Curbera, F., Meredith, G., Weerawarana, S., et al. (2001). Web services description language (wsdl) 1.1.

[Christensen, 1994] Christensen, J. (1994). Holonic manufacturing systems: initial architecture and standards directions. *Proc 1st Euro Wkshp on Holonic Manufacturing Systems*.

[Christopher, 2000] Christopher, M. (2000). The agile supply chain: competing in volatile markets. *Industrial marketing management*, 29(1):37–44.

[Colombo et al., 2005] Colombo, A., Jammes, F., Smit, H., Harrison, R., Lastra, J., and Delamer, I. (2005). Service-oriented architectures for collaborative automation. In *Industrial Electronics Society, 2005. IECON 2005. 31st Annual Conference of IEEE*, page 6 pp.

[Corba, 1995] Corba, O. (1995). Common object request broker architecture.

[Da Cunha et al., 2006] Da Cunha, C., Agard, B., and Kusiak, A. (2006). Data mining for improvement of product quality. *International Journal of Production Research*, 44(18-19):4027–4041.

[Dankbaar, 2007] Dankbaar, B. (2007). Global sourcing and innovation: the consequences of losing both organizational and geographical proximity. *European Planning Studies*, 15(2):271–288.

[Dayan, 2002] Dayan, P. (2002). Reinforcement learning. *Stevens' Handbook of Experimental Psychology*.

[De Leeuw and Volberda, 1996] De Leeuw, A. and Volberda, H. (1996). On the concept of flexibility: a dual control perspective. *Omega*, 24(2):121 — 139.

[De Masi, 2000] De Masi, D. (2000). *A sociedade Pós Industrial*. SENAC, São Paulo.

[Dewan et al., 2000] Dewan, R., Jing, B., and Seidmann, A. (2000). Adoption of internet-based product customization and pricing strategies. In *System Sciences, 2000. Proceedings of the 33rd Annual Hawaii International Conference on*, pages 10–pp. IEEE.

[Dietterich and Flann, 1997] Dietterich, T. and Flann, N. (1997). Explanation-based learning and reinforcement learning: A unified view. *Machine Learning*, 28(2):169–210.

[Dornfeld, 2010] Dornfeld, D. (2010). Sustainable manufacturing–greening processes, systems and products. *Proc. ICMC Sustainable Production for Resource Efficiency and Ecomobility, Fraunhofer Institute for Machine Tools and Forming Technology, Chemnitz University of Technology, Chemnitz*.

[Dove, 2005] Dove, R. (2005). Fundamental principles for agile systems engineering. Hoboken, NJ.

[ElMaraghy, 2005] ElMaraghy, H. (2005). Flexible and reconfigurable manufacturing systems paradigms. *International Journal of Flexible Manufacturing Systems*, 17:261–276. 10.1007/s10696-006-9028-7.

[Erl, 2006] Erl, T. (2006). *Service-oriented architecture: concepts, technology, and design*. Pearson Education India.

[Evgeniou, 2002] Evgeniou, T. (2002). Information integration and information strategies for adaptive enterprises. *European Management Journal*, 20(5):486 — 494.

[Fayyad et al., 1996] Fayyad, U., Piatetsky-Shapiro, G., and Smyth, P. (1996). From data mining to knowledge discovery in databases. *AI magazine*, 17(3):37.

[Force, 1998] Force, I. (1998). Geram: Generalised enterprise reference architecture and methodology. *version*, 1(2):30.

[Ford and Crowther, 1988] Ford, H. and Crowther, S. (1988). Today and tomorrow.

[Frank and Redmond, 1997] Frank, E. and Redmond, I. (1997). Dcom: Microsoft distributed component object model. *Redmond III November*.

[Frei et al., 2007] Frei, R., Barata, J., and Onori, M. (2007). Evolvable production systems context and implications. In *Industrial Electronics, 2007. ISIE 2007. IEEE International Symposium on*, pages 3233–3238. IEEE.

[Gardner and Bieker, 2000] Gardner, M. and Bieker, J. (2000). Data mining solves tough semiconductor manufacturing problems. In *Proceedings of the sixth ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 376–383. ACM.

[Goldman et al., 1995] Goldman, S., Nagel, R., and Preiss, K. (1995). Agile competitors and virtual organizations.

[Goranson, 1999] Goranson, H. (1999). *The agile virtual enterprise: cases, metrics, tools*. Greenwood Publishing Group.

[Gross et al., 1996] Gross, D. et al. (1996). *Forbes greatest business stories of all time*. J. Wiley & Sons.

[Guarino and Welty, 2002] Guarino, N. and Welty, C. (2002). Evaluating ontological decisions with ontoclean. *Communications of the ACM*, 45(2):61–65.

[Gutowski et al., 2006] Gutowski, T., Dahmus, J., and Thiriez, A. (2006). Electrical energy requirements for manufacturing processes. In *13th CIRP International Conference on Life Cycle Engineering*, volume 31.

[Hajarnavis and Young, 2008] Hajarnavis, V. and Young, K. (2008). An investigation into programmable logic controller software design techniques in the automotive industry. *Assembly Automation*, 28(1):43–54.

[Hames, 1994] Hames, R. (1994). *The Management Myth: Exploring the Essence of Future Organizations*. Business & Professional Publishing.

[Han and Kamber, 2006] Han, J. and Kamber, M. (2006). *Data mining: concepts and techniques*. Morgan Kaufmann.

[Harding et al., 2006] Harding, J., Shahbaz, M., Kusiak, A., et al. (2006). Data mining in manufacturing: a review. *Journal of Manufacturing Science and Engineering*, 128:969.

[Hitt et al., 1998] Hitt, M., Keats, B., and DeMarie, S. (1998). Navigating in the new competitive landscape: Building strategic flexibility and competitive advantage in the 21st century. *The Academy of Management Executive (1993-2005)*, pages 22 — 42.

[Hsieh, 2009] Hsieh, W. (2009). Machine learning methods in the environmental sciences. *Cambridge Univ. Pr., Cambridge*.

[Huyet, 2006] Huyet, A. (2006). Optimization and analysis aid via data-mining for simulated production systems. *European journal of operational research*, 173(3):827–838.

[Irani et al., 1993] Irani, K., Cheng, J., Fayyad, U., and Qian, Z. (1993). Applying machine learning to semiconductor manufacturing. *iEEE Expert*, 8(1):41–47.

[Jain and Elmaraghy, 1997] Jain, A. and Elmaraghy, H. (1997). Production scheduling/rescheduling in flexible manufacturing. *International Journal of Production Research*, 35(1):281–309.

[James-Moore, 1996] James-Moore, S. (1996). Agility is easy, but effective agile manufacturing is not. Cornfield, UK. IEEE Xplore.

[Jammes et al., 2005] Jammes, F., Mensch, A., and Smit, H. (2005). Service-oriented device communications using the devices profile for web services. In *Proceedings of the 3rd international workshop on Middleware for pervasive and ad-hoc computing*, pages 1–8. ACM.

[Josuttis, 2007] Josuttis, N. (2007). *SOA in Practice*. O'reilly.

[Jovane et al., 2009] Jovane, F., Westkämper, E., and Williams, D. (2009). *The manufuture road: towards competitive and sustainable high-adding-value manufacturing*, volume 10. Springer Verlag.

[Juehling et al., 2010] Juehling, E., Torney, M., Herrmann, C., and Droeder, K. (2010). Integration of automotive service and technology strategies. *CIRP Journal of Manufacturing Science and Technology*, 3(2):98–106.

[Kidd, 1995] Kidd, P. (1995). *Agile manufacturing: forging new frontiers*. Addison-Wesley Longman Publishing Co., Inc.

[Koestler, 1968] Koestler, A. (1968). The ghost in the machine.

[Kohavi et al., 1995] Kohavi, R. et al. (1995). A study of cross-validation and bootstrap for accuracy estimation and model selection. In *International joint Conference on artificial intelligence*, volume 14, pages 1137–1145. Lawrence Erlbaum Associates Ltd.

[Kohli and Jaworski, 1990] Kohli, A. and Jaworski, B. (1990). Market orientation: The construct, research propositions, and managerial implications. *The Journal of Marketing*, pages 1 — 18.

[Komoda, 2006] Komoda, N. (2006). Service oriented architecture (soa) in industrial systems. In *Industrial Informatics, 2006 IEEE International Conference on*, pages 1–5. IEEE.

[Koren, 2010] Koren, Y. (2010). *The Global Manufacturing Revolution: Product-Process-Business Integration and Reconfigurable Systems*, volume 75. Wiley.

[Koren et al., 1999] Koren, Y., Heisel, U., Jovane, F., Moriwaki, T., Pritschow, G., Ulsoy, G., and Van Brussel, H. (1999). Reconfigurable manufacturing systems. *CIRP Annals-Manufacturing Technology*, 48(2):527–540.

[Kornhauser, 1959] Kornhauser, W. (1959). *The politics of mass society*. Free Press New York.

[Kotsiantis et al., 2007] Kotsiantis, S., Zaharakis, I., and Pintelas, P. (2007). Supervised machine learning: A review of classification techniques. *Frontiers in Artificial Intelligence and Applications*, 160:3.

[Kubat et al., 1998] Kubat, M., Holte, R., and Matwin, S. (1998). Machine learning for the detection of oil spills in satellite radar images. *Machine learning*, 30(2):195–215.

[Lastra and Delamer, 2006] Lastra, J. and Delamer, M. (2006). Semantic web services in factory automation: fundamental insights and research roadmap. *Industrial Informatics, IEEE Transactions on*, 2(1):1–11.

[Lee et al., 2011] Lee, J., Ghaffari, M., and Elmeligy, S. (2011). Self-maintenance and engineering immune systems: Towards smarter machines and manufacturing systems. *Annual Reviews in Control*.

[Leitão, 2004] Leitão, P. (2004). An agile and adaptive holonic architecture for manufacturing control.

[Levitt, 1993] Levitt, T. (1993). The globalization of markets. *Readings in international business: a decision approach*, 249.

[Lewis, 1998] Lewis, R. W. (1998). *Programming Industrial Control Systems Using Iec 1131-3*. IET.

[Li and Olafsson, 2005] Li, X. and Olafsson, S. (2005). Discovering dispatching rules using data mining. *Journal of Scheduling*, 8(6):515–527.

[Liao, 2005] Liao, S. (2005). Expert system methodologies and applications—a decade review from 1995 to 2004. *Expert systems with applications*, 28(1):93–103.

[Lucas, 2003] Lucas, M. R. (2003). *Understanding and assessing logic control design methodologies*. PhD thesis, University of Michigan.

[Mathes et al., 2009] Mathes, M., Stoidner, C., Heinzl, S., and Freisleben, B. (2009). Soap4plc: web services for programmable logic controllers. In *Parallel, Distributed and Network-based Processing, 2009 17th Euromicro International Conference on*, pages 210–219. IEEE.

[Mehrabi et al., 2002] Mehrabi, M. G., Ulsoy, A. G., Koren, Y., and Heytler, P. (2002). Trends and perspectives in flexible and reconfigurable manufacturing systems. *Journal of Intelligent Manufacturing*, 13(2):135–146.

[Michalski et al., 1983] Michalski, R., Carbonell, J., and Mitchell, T. (1983). Machine learning: an artificial intelligence approach.

[Michie et al., 1994] Michie, D., Spiegelhalter, D., Taylor, C., and Campbell, J. (1994). Machine learning, neural and statistical classification.

[Minton et al., 1989] Minton, S., Carbonell, J., Knoblock, C., Kuokka, D., Etzioni, O., and Gil, Y. (1989). Explanation-based learning: A problem solving perspective. *Artificial Intelligence*, 40(1):63–118.

[Mitchell, 1991] Mitchell (1991). *CIM Systems: an introduction to computer-integrated manufacturing*. Prentice-Hall Inc.

[Mitchell et al., 1993] Mitchell, T., Thrun, S., et al. (1993). Explanation-based neural network learning for robot control. *Advances in Neural information processing systems*, pages 287–287.

[Mitchell, 1997] Mitchell, T. M. (1997). Machine learning. 1997. *Burr Ridge, IL: McGraw Hill*, 45.

[Monostori, 2003] Monostori, L. (2003). Ai and machine learning techniques for managing complexity, changes and uncertainties in manufacturing. *Engineering Applications of Artificial Intelligence*, 16(4):277–291.

[Monostori et al., 1996] Monostori, L., Márkus, A., Van Brussel, H., and Westkamp, E. (1996). Machine learning approaches to manufacturing. *CIRP Annals-Manufacturing Technology*, 45(2):675–712.

[Morrison Paul and Siegel, 2001] Morrison Paul, C. and Siegel, D. (2001). The impacts of technology, trade and outsourcing on employment and labor composition. *The Scandinavian Journal of Economics*, 103(2):241–264.

[Murray and Golluscio, 2002] Murray, P. and Golluscio, E. (2002). Corba and web services. *Cape Clear Software–2002 http://www. omg. org/news/whitepapers/CORBA-Web% 20Services. pdf (acessado em Janeiro 2006)*.

[Nambiar, 2009] Nambiar, A. (2009). Agile manufacturing: A taxonomic framework for research. IEEE.

[Narula, 2003] Narula, R. (2003). *Globalization & technology: interdependence, innovation systems and industrial policy*. Polity.

[Narver and Slater, 1990] Narver, J. and Slater, S. (1990). The effect of a market orientation on business profitability. *The Journal of Marketing*, pages 20–35.

[Natis, 2003] Natis, Y. V. (2003). Service-oriented architecture scenario.

[Negroponte, 1996] Negroponte, N. (1996). *Being Digital*. Vintage.

[Neves and Barata, 2009] Neves, P. and Barata, J. (2009). Evolvable production systems. In *Assembly and Manufacturing, 2009. ISAM 2009. IEEE International Symposium on*, pages 189–195. IEEE.

[Nilsson, 1996] Nilsson, N. (1996). Introduction to machine learning.

[Noaker, 1994] Noaker, P. (1994). The search for agile manufacturing.

[Noble, 2011] Noble, D. (2011). *Forces of production: A social history of industrial automation*. Transaction Publishers.

[Okino, 1993] Okino, N. (1993). Bionic manufacturing systems. In *Conference on Flexible Manufacturing Systems, Past, Present-Future (Ed: J. Peklenik), Ljubljana: Faculty of Mechanical Engineering*.

[Oleson, 1998] Oleson, J. (1998). *Pathways to agility: mass customization in action*. Wiley.

[Onori, 2002] Onori, M. (2002). Evolvable assembly systems-a new paradigm? In *33rd Int. Symposium on Robotics (ISR)*, pages 617–621.

[Papazoglou and van den Heuvel, 2007] Papazoglou, M. and van den Heuvel, W.-J. (2007). Service oriented architectures: approaches, technologies and research issues. *The VLDB Journal*, 16(3):389–415.

[Pietrusewicz and Urbanski, 2011] Pietrusewicz, K. and Urbanski, L. (2011). Control programming software strategies for industrial systems. *Control Engineering*.

[Ranky, 1990] Ranky, P. (1990). *Flexible manufacturing cells and systems in CIM*. CIMware.

[Rao, 2010] Rao, R. (2010). *Advanced Modeling and Optimization of Manufacturing Processes: International Research and Development*. Advanced Manufacturing. Springer.

[Reyes et al., 2002] Reyes, A., Yu, H., Kelleher, G., and Lloyd, S. (2002). Integrating petri nets and hybrid heuristic search for the scheduling of fms. *Computers in Industry*, 47(1):123–138.

[Ribeiro and Barata, 2011] Ribeiro, L. and Barata, J. (2011). Re-thinking diagnosis for future automation systems: An analysis of current diagnostic practices and their applicability in emerging IT based production paradigms. *Computers in Industry*.

[Ribeiro et al., 2008] Ribeiro, L., Barata, J., and Mendes, P. (2008). MAS and SOA: complementary automation paradigms. In Azevedo, A., editor, *Innovation in Manufacturing Networks*, volume 266 of *IFIP International Federation for Information Processing*, pages 259–268. Springer Boston.

[Ribeiro et al., 2011] Ribeiro, L., Candido, G., Barata, J., Schuetz, S., and Hofmann, A. (2011). It support of mechatronic networks: A brief survey. In *Industrial Electronics (ISIE), 2011 IEEE International Symposium on*, pages 1791–1796. IEEE.

[Romanowski and Nagi, 2001] Romanowski, C. and Nagi, R. (2001). Analyzing maintenance data using data mining methods. In *Data mining for design and manufacturing*, pages 235–254. Kluwer Academic Publishers.

[Rossi and Dini, 2000] Rossi, A. and Dini, G. (2000). Dynamic scheduling of fms using a real-time genetic algorithm. *International Journal of Production Research*, 38(1):1–20.

[Scheer, 1991] Scheer, A.-W. (1991). *CIM: Computer Integrated Manufacturing: towards the factory of the future*. Springer, Berlin.

[Schmitt et al., 2011] Schmitt, R., Bittencourt, J., and Bonefeld, R. (2011). Modelling machine tools for self-optimisation of energy consumption. *Glocalized Solutions for Sustainability in Manufacturing*, pages 253–257.

[Scholten, 2007] Scholten, B. (2007). *The road to integration: A guide to applying the ISA-95 standard in manufacturing*. Isa.

[Schwarz, 2005] Schwarz, K. (2005). The standard message specification for industrial automation systems - ISO 9506 (MMS). In *The IndustrialInformation Technology*, pages 898 – 929. CRC Press, San Francisco.

[Seabra Lopes and Camarinha-Matos, 1995] Seabra Lopes, L. and Camarinha-Matos, L. (1995). A machine learning approach to error detection and recovery in assembly. In *Intelligent Robots and Systems 95.'Human Robot Interaction and Cooperative Robots', Proceedings. 1995 IEEE/RSJ International Conference on*, volume 3, pages 197–203. IEEE.

[Self-Learning, 2010a] Self-Learning (2010a). Deliverable 1.2 self-learning concept wp 100.

[Self-Learning, 2010b] Self-Learning (2010b). Deliverable 1.3 self-learning public concept wp 100.

[Self-Learning, 2010c] Self-Learning (2010c). Deliverable 1.5 business cases infrastructure specification wp 500.

[Shen et al., 2006] Shen, W., Onori, M., Barata, J., and Frei, R. (2006). Evolvable assembly systems basic principles. In *Information Technology For Balanced Manufacturing Systems*, volume 220 of *IFIP International Federation for Information Processing*, pages 317–328. Springer Boston.

[Simon, 1980] Simon, H. (1980). *Why should machines learn?* Carnegie-Mellon University.

[Srinivas and Shahbaz, 2004] Srinivas, J. and Shahbaz, M. (2004). Agent oriented planning using data mined knowledge. *10th International Conference on Concurrent Engineering, Adaptive Engineering for Sustainable Value Creation*, pages 301 –307.

[Stecke, 1985] Stecke, K. (1985). Design, planning, scheduling, and control problems of flexible manufacturing systems. *Annals of Operations Research*, 3(1):1–12.

[Tapscott, 2009] Tapscott, D. (2009). *Grown up digital*. McGraw-Hill New York.

[Teece, 2009] Teece, D. (2009). *Dynamic capabilities and strategic management: organizing for innovation and growth*. Oxford University Press, USA.

[Tharumarajah, 1996] Tharumarajah, A. (1996). Comparison of the bionic, fractal and holonic manufacturing system concepts. *International Journal of Computer Integrated Manufacturing*, 9(3):217–226.

[Thrun, 1996] Thrun, S. (1996). Learning to learn: Introduction. In *In Learning To Learn*. Citeseer.

[Toffler, 1980] Toffler, A. (1980). *The third wave*. Morrow.

[Tsatsoulis and Kashyap, 1993] Tsatsoulis, C. and Kashyap, R. (1993). Case-based reasoning and learning in manufacturing with the toltec planner. *Systems, Man and Cybernetics, IEEE Transactions on*, 23(4):1010–1023.

[Turban et al., 2005] Turban, E., Aronson, J., and Liang, T. (2005). *Decision Support Systems and Intelligent Systems 7 Edition*. Pearson Prentice Hall.

[Uddin et al., 2011] Uddin, M., Dvoryanchikova, A., Lastra, J., Scholze, S., Stokic, D., Candido, G., and Barata, J. (2011). Service oriented computing to self-learning production system. In *Industrial Informatics (INDIN), 2011 9th IEEE International Conference on*, pages 212–217. IEEE.

[Van Brussel et al., 1998] Van Brussel, H., Wyns, J., Valckenaers, P., Bongaerts, L., and Peeters, P. (1998). Reference architecture for holonic manufacturing systems: PROSA. *Computers in industry*, 37(3):255–274.

[Vapnik, 1996] Vapnik, V. (1996). *The nature of statistical learning theory*. Springer-Verlag, New York.

[Waldner, 1992] Waldner, J. (1992). *CIM: principles of computer integrated manufacturing*. Wiley.

[Wirth and Hipp, 2000] Wirth, R. and Hipp, J. (2000). Crisp-dm: Towards a standard process model for data mining. In *Proceedings of the 4th International Conference on the Practical Applications of Knowledge Discovery and Data Mining*, pages 29–39. Citeseer.

[Witten and Frank, 2005] Witten, I. and Frank, E. (2005). *Data Mining: Practical machine learning tools and techniques*. Morgan Kaufmann.

[Womack et al., 1990] Womack, J., Jones, D., and Roos, D. (1990). *The machine that changed the world*. Harper Perennial, New York.

[Yuan et al., 2010] Yuan, R., Li, Z., Guan, X., and Xu, L. (2010). An svm-based machine learning method for accurate internet traffic classification. *Information Systems Frontiers*, 12(2):149–156.

[Yusuf et al., 1999] Yusuf, Y. Y., Sarhadi, M., and Gunasekaran, A. (1999). Agile manufacturing:: The drivers, concepts and attributes. *International Journal of Production Economics*, 62(1):33–43.

[Zhang and Sharifi, 2007] Zhang, Z. and Sharifi, H. (2007). Towards theory building in agile manufacturing strategy—a taxonomical approach. *Engineering Management, IEEE Transactions on*, 54(2):351 — 370.

[Zhu, 2005] Zhu, X. (2005). Semi-supervised learning literature survey.

[Zoitl and Vyatkin, 2009] Zoitl, A. and Vyatkin, V. (2009). IEC 61499 architecture for distributed automation: the "Glass half full" view. *Industrial Electronics Magazine*, pages 7 — 23.